University of Manchester

School of Computer Science
Project Report 2014

Using Machine Learning to Predict Personal Expenditure

Author:
Pez Cuckow

Supervisor:
Dr Gavin Brown

Abstract

A complete understanding of personal finances is becoming increasingly important as the average persons disposable income has decreased due to a changing financial climate.

The aim of this project is to build an application that makes it easier to manage a users personal finances. This is split into two halves, accessing historical information in an easy to understand way and using machine learning techniques to predict future financial transactions. The security considerations of storing personal finance information are also considered.

This begins with a review of the existing commercial personal finance applications and the current techniques used to forecast time-boxed financial data, such as the value of a stock on the stock market, before detailing the design and implementation of the application.

Having completed the application, the performance of selected techniques is reviewed, before discussing further research opportunities which could improve the applications accuracy.

Project Title: Using Machine Learning to Predict Personal Expenditure
Author: Pez Cuckow
Degree: Computer Science with Business and Management
Supervisor: Dr Gavin Brown

Keywords: Markov Chain Models, Weighted Arithmetic Mean, Responsive Web Design, Web System Security

Contents

List of Figures

List of Tables

Glossary

category transactors have a category and a subcategory, e.g. Tesco = Shopping, Groceries. 13, 14, 19

global transactor the system holds two collections of transactors and mapping's, the global ones are shared between all users. 22, 39

mapping this connects the reference found on a statement to a Transactor. e.g. both Snbs and Sains represent Sainsbury's. 22

reference the memo or message that is included on the bank statement with a transaction. 10, 22, 23

transaction a single movement of money from/to a Transactor. 9, 10, 69

transactor somewhere money is spent, e.g. Tesco, Sainsbury's, Byte Cafe.. 21, 39

user transactor the user transactors and mapping's are unique to each user. 22, 39

Chapter 1

Introduction

Traditionally the management of personal finances is performed by viewing bank statements provided by the users bank. In the modern age of 'Internet banking', banks offer a limited set of tools that mimic the paper statements seen historically.

This project sets out to build an online application that can be used to manage personal finances. There are two main parts of the project; firstly users can upload bank statements, which are displayed and navigated in an intuitive manner; secondly, once the application has enough historical data, predicting the users future outflow.

1.1 Motivation

There are four main steps when producing and using a budget: recording previous expenses, sorting these into categories, using this historical information to estimate future expenditure, and evaluating the accuracy of predictions based on the new information and adjusting accordingly.

Since the liquidity crisis of 2009 [1], budgets have been squeezed and the average personal disposable income has fallen significantly, hitting a nine-year low in 2012 [2]. Experts suggest that "Budgets are essential for financial planning" [3], research suggesting that personal budgets lead to a "positive impact" on "mental wellbeing" [4] and guides from UCAS, the UoM SU Advice Centre and The Manchester University Crucial Guide encouraging use of budgeting, it is clear that producing a budget is of benefit.

In an informal survey[1] by the researchers, however, the majority of students questioned did not heed this advice, and were not following a budget. Producing an easier way to manage personal finances and predict future outflow can hopefully reduce the barriers to entry for creating budgets and increase the people using one.

Increasing use of debit cards [5] means that bank statements contain more and more information about where people spend their money. With access to those bank statements now provided online, and most UK banks offering the option to export transaction history, individual users can collate a database of their personal spending habits.

[1] Appendix A

The increasing availability of this data, combined with more detailed transaction history makes it potentially possible to automate the four main steps of producing a budget, and this is the main objective of the project.

1.2 Aims and Objectives

The key objectives of this project can be split into three parts, the management of statements, making predictions of future outflow using those statements and ensuring a high level of security.

1.2.1 Statement Management

Implement an intuitive way to view and manage personal finances. There are several key parts to this, upload and parsing of transactions from statements downloaded from the users bank, resolving the references found on the statement to the real world business they represent and categorising the individual transactions to make them easier to understand.

1.2.2 Prediction

Accurately predicting a users future transactions based on their transactions history. The prediction should be made using a model that is fitted to each users individual spending patterns, and is evaluated in order to improve the model. The application will need to predict whether or not spending will occur and how much money will be spent.

1.2.3 Security

The project should be secure and uphold the high security expectations of users uploading their personal information. The application will deal with information of a sensitive nature, therefore strong security techniques are of high importance to ensure no loss of personally identifiable information. The project should take this into account, considering possible attack vectors and taking steps to mitigate those attacks.

1.3 Overview of Report

Chapter 2 reviews existing commercial personal finance applications and existing techniques used to forecast time-boxed financial information.

Chapter 3 details the key design decisions made when planning the software and how the techniques in the background research are applied.

Chapter 4 includes implementation specifics of some of the features outlined in the design, focused particularly on features that were difficult to implement.

Chapter 5 gives an outline of the finished applications features through the use of screenshots in a walkthrough.

Chapter 6 reviews the application in terms of it's user experience, prediction accuracy and through the use of penetration testing.

The report concludes in chapter 7 which compares the project aims to what was achieved, suggests further enhancements that could be added to the application and outlines some further research areas.

Chapter 2

Background

As outlined in chapter 1 the project consists of two major parts: a financial management service, which can be used to view historical spending and gain understanding of personal finances; and a forecasting element which predicts how much spending will occur in the future.

2.1 Statement Management

There are existing applications that implement similar features to the money management aims of this project. The most basic examples of a similar applications are the majority of Internet banking services in the UK. These services set out to implement statements that behave in the same way hard copy paper statements are used, a customer can view their statements organised by month and their current balance, but provide no additional functionality such as organising the transactions or customising the view of the statement. However, more advanced Internet banking services do exist, most notably Money Manager offered by Lloyds TSB, the first and only personal money management application provided by a UK bank and Mint.com a United States (US) only personal finance service [6], [7].

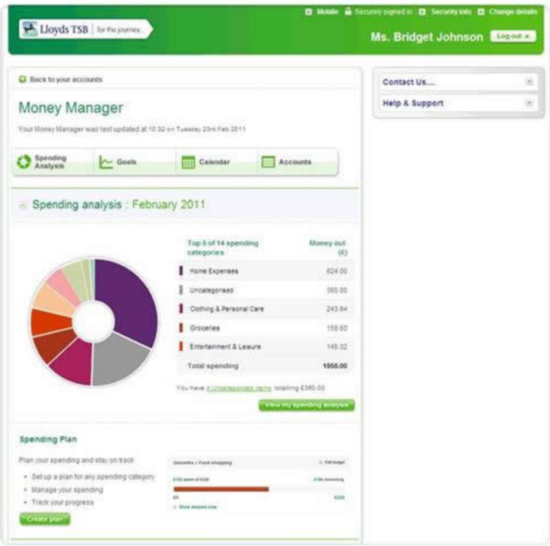
2.1.1 Lloyds Money Manager

The service is available to Lloyds TSB current account holders as part of their online banking and its features revolve around documenting historical spending [8].

The key features include:

- Categorising spending
- Creating spending plans per category
- Viewing money spent per category
- Track progress of budget targets

Customer reviews of the service highlight the usefulness of spending analysis screen, which includes a breakdown of spending in each category (Fig. 2.1, as well as the spending calendar, which displays money spent in a day by day format. The reviews, highlight some shortcomings; noting that changes to categories are not reflected immediately, categories are often incorrect and that it's not possible to override the category for a single transaction, for example food bought at a petrol station is placed in the Car category and cannot be moved [9], [10].



Your Money Manager was last updated at 10:32 on Tuesday 23rd Feb 2011

Top 5 of 14 spending categories	Money out (£)
Home Expenses	624.00
Uncategorised	360.00
Clothing & Personal Care	243.94
Groceries	158.60
Entertainment & Leisure	145.32
Total spending	1950.00

Plan your spending and stay on track

- Set up a plan for any spending category
- Manage your spending
- Track your progress

Figure 2.1: Spending Analysis by category on Lloyds Money Manager [8]

A key advantage of the money manager is that Lloyds already have access to their customer data no data entry or upload is required, which could be confusing and off-putting to potential users.

2.1.2 Mint.com

Mint.com offers very similar features to Lloyds but is limited to the US. However, Mint automatically logs into the users online bank account and downloads their statements authenticating with their banking username and password. It's reported that this feature relies on the use of application programming interface's (API) at each bank which Intuit (the company behind Mint) have negotiated access individually, though Intuit have published no information to support or dispute this [11], [12]. Although this feature is clearly useful and saves time for the users, it does make Mint responsible for storing their customers Internet banking passwords and presumably involves fee payments to the banks providing these API's. For these reasons it was decided that automatic statement uploading was outside of the budget and scope of this project, however, the project should support manual upload of statements to avoid data entry by users.

2.1.3 Mobile Apps

Mobile applications (or 'apps') have seen a surge in popularity since the release of smartphones and are a widespread target for small pieces of software, such as financial organisation [13].

The three most popular iPhone personal financial applications [14], at the time of planning the project, all offered features very similar to those found in the Lloyds Money Manager and Mint. The most popular features being grouping money by category and graphs of spending history. However, they all had the same drawback, the user had to manually enter all of their transactions and set categories for them, which appears time consuming and error prone, particularly on a mobile app [15]–[17].

The increase of mobile usage should be considered when planning the features of the project, with the project ensuring mobile compatibility and if possible, avoiding manual data entry.

2.2 Prediction

Predicting future transactions before they occur is technically similar to the work done by investors on the stock market, where the objective is to predict whether the value of a stock will fall or increase in order to make buy/sell decisions.

Preifer and Carraway demonstrated that Markov Chain Models can be used to model customer relationships with a business and predict the expected value of a marketing engagement with an individual customer. By creating a transition matrix of a particular customer transitioning from not spending to spending and visa versa over five periods[1], they were able to estimate the likelihood of a spend occurring in a given period, Fig. 2.2 shows a graphical representation of the model that was produced, the states represent the five periods, where p_i is the probability of the transition occurring during period i [18]. The researchers calculated the expected loan to value ratio (LTV) for the customer over the periods, by producing a matrix of costs and gains associated with a purchase in each period and multiplying that by the probability of a purchase occurring taken from the transition matrix. This gives the expected present value for each period, which can be used to decide when to end a relationship with a customer (preventing the costs). They demonstrate the application of Markov Chain Models (MCM's) to a larger dataset, calculating the optimal policy for ending relationships with customers depending on varying costs concluding that the use of MCM's is an effective way of making customer relationship decisions. However, this paper assumes the company making predictions already knows how much money a customer will spend during each interaction and is focused around calculating the probability of a spend occurring. An implementation applied to the personal spending space will require a way to predict the value of the future transaction.

A MCM is a mathematical model that is defined by a collection of states and the set of probabilities of a transition occurring between those states. The model visualised as a

[1] An 'illustration' assuming a customer will never return after 5 months of not spending

graph shown in Fig. 2.2 defines five states, and shows the probability of the transitions occurring those states on the edges. As the likely hood of transitions are known, the probability of a particular sequence occurring can be calculated, for example the probability of the sequence 11234 in the figure is $p(11234) = p(1)p(1|1)p(2|1)p(3|2)p(4|3)$.

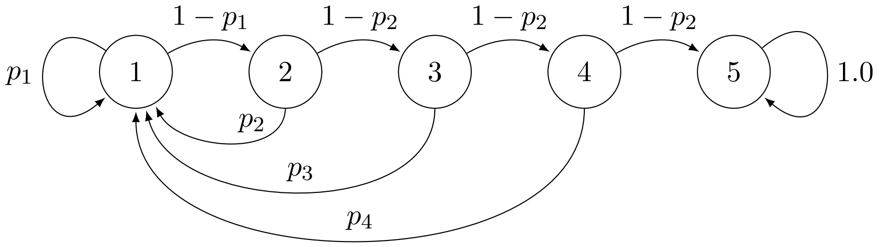


Figure 2.2: Markov Chain Model for a particular customer over five periods [18, Adapted from Fig. 1]

Research by Singh et al., from the Massachusetts Institute of Technology, studied the spending behaviour of 52 adults and investigated the impact of social interactions, including text messages, phone calls and face-to-face meetings, on the participants spending in order to predict their spending behaviour. Using a Naïve Bayes classifier and selecting a subset of their available features using an Information Gain approach, choosing those with most relevance to each classification task, they were able to correctly classify whether the participant would overspend, explore a diverse range of businesses and remain loyal to a business with 72% overall accuracy. They concluded that social factors, were better "predictors of spending behaviour" than personality traits, which had been previously studied [19]. Although this paper did not study the affects of the participants previous transactions on spending, they were able to predict the users spending behaviour, highlighting that factors other than the transaction history may be of importance when trying to predict a users future outflow. However, the paper does not attempt to make a prediction of the amount spent or how many transactions occur.

Smoothing is typically applied to financial market data, for example the value of a particular stock on the FTSE 100. The most common techniques are simple, weighted and exponential moving averages, which all reduce the noise found in the data potentially revealing an orderly process, by removing outliers found in the data. The result of this effect can be seen in Fig 2.3 [20].

These techniques can be applied to a discrete set of numerical time series data, such as personal expenditure over time in order to make an estimate of what the next value in the series will be [21]. A prediction can be made using the formula in Fig. 2.4, where w_i is the weight and x_i is the value at time period i. Simple smoothing is the equivalent of $w_i = 1$, while exponential smoothing is based around a negative exponential law such as $w_i = e^{-n+i}$, both are examples of weighted smoothing and the weights can be decided

[2] A stock market index of 500 American companies, the US equivalent of the FTSE 500

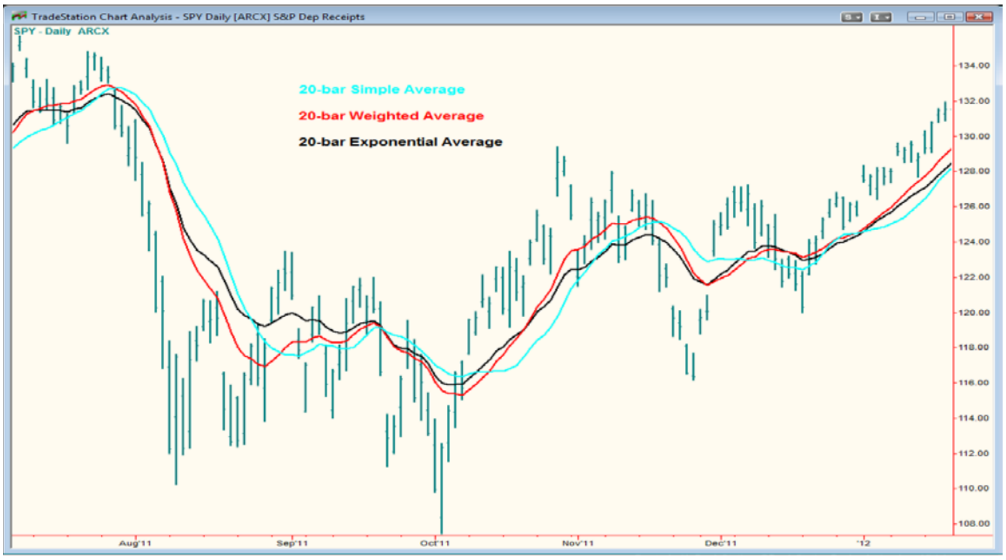


Figure 2.3: Simple Moving Average (MA) [blue], Weighted MA [red] and Exponential MA [black] of the S&P500[2][20, Fig. 5]

in different ways depending on what is being predicted. Time periods with a higher weight have a greater affect on the mean, so in order to make a future prediction, the most recent time period would have a higher weight.

$$\frac{w_1x_1 + w_2x_2 + \cdots + w_nx_n}{w_1 + w_2 + \cdots + w_n}.$$

Figure 2.4: Using weighted smoothing to predict a future value

Smoothing (and therefore prediction) can be extended to take into account trends and possible seasonal fluctuations using double and triple smoothing, respectively. A technique known as ‘Holt-Winters double exponential smoothing‘ takes into account trends in data, which single smoothing does perform accurately with, by factoring the weighted average growth between previous time series when calculating the average for each period [22]. Extending the calculation into double and triple smoothing when estimating a users future outflow was decided as a possible extension for the project. A discussion of this technique can be found in subsection 7.2.2.

2.3 Security

As the project stores peoples personally identifiable information and needs to be backed by strong security standards, before designing the project and as part of the ethics application procedure common web security practice was researched.

2.3.1 Account Hijacking

On the web, information is sent between the users web browser and the remote server over HTTP information in plain text and can easily be read using a man-in-the-middle attack. This risk is compounded if accessing the Internet via an an unencrypted WiFi connection[3] which would allow anyone in the local area to 'sniff' the information by simply scanning for capturing the transmitted packet.

If a website involves authentication, this becomes a serious security risk. Authentication is usually performed by sending the username and password in plain text to a remote server which is validated, and if valid issuing the user with a session cookie. A potential attacker could observe and store these usernames and passwords, which is why commonly used websites such as Facebook use HTTPS for the login, ensuring the usernames and passwords are sent encrypted to the server.

If a website falls back to HTTP following authentication, the risk of unauthorised account access is still prevalent. In order to identify which user the browser is authenticated as it sends a session cookie to the remote server with each request. Although the attacker can't observe the username and password, the session cookie is being sent in plain text with every request, and the attacker can perform a session hijack by downloading the content of that cookie to their local machine (Fig. 3.12) and then sending it to the remote server with their HTTP request 'proving' they are the user and gaining access to their account (Fig. 3.13) [23]. Firesheep was a proof of concept plugin for Firefox released in 2010 that demonstrated this vulnerability, showing that session hijacking could be performed on popular sites including Google, Facebook, Twitter, Dropbox and Flickr, which until recently were not using sitewide SSL to protect their cookies [24], [25]. More recently 'WhatsApp Sniffer', available on Google Play until May 2012 was able to display messages addressed to other WhatsApp users connected to the same network using this technique [26].

2.3.2 Password Security

A common cracking technique used to gain unauthorised access to a user account is a known as a brute force attack. If an attacker knows a particular users username they can perform targeted guessing of the password by enumerating through all possibilities. A websites ability to resist this kind of attack is called the 'password guessing resistance'. It is for this reason that many websites enforce password rules in an attempt to increase the number of possible combinations for a password, or the entropy [27].

Shannon Entropy can be used estimate the strength of a passwords resistance to this kind of attack. The entropy is calculated using $H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$ where $p(x_i)$ is the probability of the value x occurring [28]. The paper suggests a predefined set of rules for estimating entropy based on Shannon's work studying English text, however other papers found that using this predefined set of rules was not a valid measure of password strength [29].

[3] Common at Coffee Shops and Universities

2.3.3 Database Storage

Unfortunately, it is common for the contents of a websites database to be leaked, whether by an administrator of the website or using other techniques such as SQL injection [30], [31]. It's important to think about the security of the data held within the database, as well as the security of the front end.

If a users password is stored in a reversible state, whether that is plain text or encrypted and the database is leaked, not only can the users account on the original website be compromised, but the data can be used attack other websites where the user uses the same username and password. Encryption is equivalent to plain text in that, it is simply a case of finding the encryption key, which is very possible using brute force and all the data is in plain text. This is why standard security practice is to hash passwords. The only way to 'decrypt' a hash is to guess the original input by brute force and see if that matches the output. However crackers often make use of Rainbow tables, collections of precalculated hashes and the input used to create them, allowing an attacker to simply lookup a hash in their database to get the result rather than enumerating all the possibilities [32]. To reduce the effectiveness of this attack method, 'salting' is commonly used. Salting involves adding a random collection of numbers and letters to each password. This means that the generated hash is dependent on both the users password and the salt, and therefore a Rainbow Table would need to be generated for each user, cancelling out the advantage of precalculation and making the tables useless [33].

Chapter 3

Design

This chapter covers design of the system, including an overview of the architecture and descriptions of the key components.

3.1 Statement Management

The statement management features of the application were selected based on the functionality observed during the background research and conversations with potential users, asking what features they enjoyed from their current Internet banking and what additional features they would find useful to manage their statements

The key features include; parsing of files downloaded from Internet banking, mapping transactions found in the files to real world businesses (transactors), organising transaction history by category or transactor and viewing all transactions at a particular transactor.

3.1.1 Upload

To get a users transaction history they must first upload a file containing their historical transactions.

The major UK banks tested[1] provided statement downloads in Quicken, Microsoft Money or Microsoft Excel format. Further investigation revealed that the underlying formats were Quicken Interchange Format (QIF), Open Financial Exchange (OFX) and comma-separated values (CSV).

As there is no pre-defined standard for bank statements in CSV format, upon investigation, it became clear that the banks used completely different structures. It was decided the application would parse the QIF and OFX formats, following their respective specifications. Examples of QIF and OFX can be seen in Fig. 3.2.

It was quickly identified that although the QIF/OFX files were following the same specification, depending on the bank they had different structures, and in some cases the structures even varied within the same bank, depending on the exact wording of the

[1] Natwest, First Direct and HSBC

```
% QIF FORMAT
!Type:Bank
D28-06-13
PASDA SUPERSTORE       TROWBRIDGE
T-15.00
^
D28-06-13
PPAYPAL PAYMENT
T-12.50
^
```

Figure 3.1: Two transactions in QIF format

```
% OFX FORMAT
<STMTTRN>
<TRNTYPE>POS</TRNTYPE>
<DTPOSTED>20130628</DTPOSTED>
<TRNAMT>-15.00</TRNAMT>
<NAME>ASDA SUPERSTORE</NAME>
<MEMO>TROWBRIDGE</MEMO>
</STMTTRN>
<STMTTRN>
<TRNTYPE>DEBIT</TRNTYPE>
<DTPOSTED>20130618</DTPOSTED>
<TRNAMT>-12.50</TRNAMT>
<NAME>PAYPAL PAYMENT</NAME>
</STMTTRN>
```

Figure 3.2: Two transactions in OFX format

download. Interestingly an OFX file downloaded from First Direct was found to be in QIF format, despite an .ofx suffix.

Notably there were discrepancies with the formatting of dates in QIF. The specification from Intuit[2] does not specify a date format [34]. The sample files tested included dates in D-M-Y, M-D-Y and Y-M-D format.

To combat this, three steps of resiliency were added to the design of the upload system, seen in Fig. 3.3.

Having uploaded the file the system first identifies the file type by looking inside the file and parsing its contents, ignoring the extension and rejecting the file if it matches neither format.

If the file is QIF the parser parses all transactions up front and evaluates the format

[2] The developers of Quicken and QIF

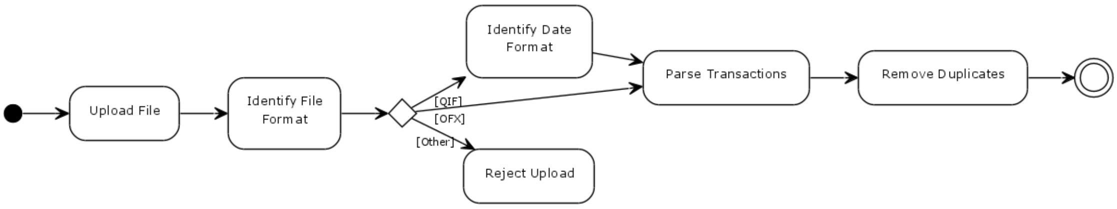


Figure 3.3: Activity diagram for statement uploads

of the dates. If the dates are found to have the format 00−00−0000 the system needs to decide whether that's DD-MM-YYYY or MM-DD-YYYY, otherwise performing standard date parsing of the string. To decide between D-M-Y or M-D-Y the application goes through all the dates and attempts to parse in both formats, if a format fails it is marked as incorrect. This leaves the application in one of the four states, seen in Table 3.1. In the case of state 1 or 4 there is ambiguity and the application prompts the user. This ambiguity can be caused by dates that are malformed or a collection of dates falling within a range that doesn't have a day value over 12, as both formats parse correctly.

State	D-M-Y	M-D-Y
1	true	true
2	true	false
3	false	true
4	false	false

Table 3.1: Possible states following evaluation of transaction dates

In provisional user testing, it was discovered that users had a tendency to upload the same file more than once or to upload statements with an overlapping date range. To account for this, before creating a new Transaction the application checks for an identical transaction for the current user in the database and if one is found, skips creating a new Transaction. For speed this is done using a stored unique value, resulting from a SHA512 hash of date posted, transaction value, transactor, memo and transaction id[3], which is generated when saving a Transaction to the database.

3.1.2 Named Entity Resolution

Almost all functionality of the project relies on successfully mapping the text found on a bank statement that represents a business or person to a single entity in the application, known as a transactor by the system. After a cleanup of different suffixes that banks append it was found that transactors are often referenced using several names.

Seen in Table 3.2, Sainsbury's was referred to nine different ways in the statement data uploaded by the research participants and similar results are found for most transactors.

[3] If a one was provided by the users bank

Reference	Occurrences
sainsburys s/mkts	46
sainsburys s/mkt	9
sainsburys s/mkts cd	7
js online grocery	2
sainsbury s/mkt cd	2
sainsburys smkt	2
js online grocer	1
sainsburys superma	1
sainsburys-superma	1

Table 3.2: References to the entity ‘Sainsbury’s’ found in participant data

Mapping to Entities

In consideration of this, the concept of mappings was added to the system. A mapping is a single reference to a transactor, such as ‘sainsbury s/mkt’. A transactor has multiple mappings. Fig. 3.4 shows this structure.

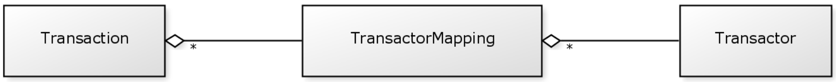


Figure 3.4: Overview of Mapping objects

Global vs User

As identified in the background research, it should be possible for users to both categorise and organise transactions according to their preferences and override existing categories, however categories chosen by a particular user should not affect other users.

To support this the application stores two sets of mappings and transactions, User and Global. The structure of the relevant objects is shown in Fig. 3.5. A Transaction can have both a UserMapping and a GlobalMapping, in which case the UserMapping overrides the GlobalMapping when calling methods such as getMapping() on the Transaction.

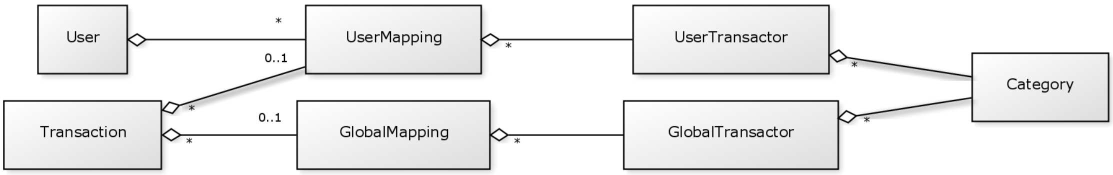


Figure 3.5: Overview of User Mappings

3.1.3 Suggestions

Having mapped references to entities, the system is able to use this knowledge to make suggestions of appropriate entities for unseen references in some cases to help streamline the naming process for potential users. This is performed by taking the list of mappings and finding those with the smallest difference to the unseen reference. Difference can be calculated in several different ways, including the Levenshtein distance which calculates the number of single-character edits to transform between the two strings, implementation details for this project can be found in Section 4.2.3 [35].

3.2 Prediction

In order to make a prediction of how much money a user will spend and receive in a given period, two steps need to be completed; predicting whether or not each individual transaction will occur in a given month; and estimating how much money will be involved.

Drawing from the research detailed in chapter 2, the system uses a First-order Markov Chain model to decide whether or not transactions will occur, and weighted arithmetic means to predict how much money will be spent.

3.2.1 Markov Chain Models

An easy way to visualise the Markov Chain Models the system is creating is through a directed graph. Two frequent examples are shown in Fig. 3.6 and 3.7, taken from participant data, where 0 represents a transaction not occurring, 1 is the opposite and the edges are labelled with the probability of transitioning from one to the other.

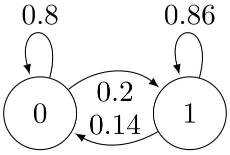


Figure 3.6: Transition diagram for a monthly pay check

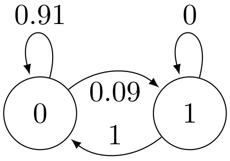


Figure 3.7: Transition diagram for a one off purchase

3.2.2 Weighted Arithmetic Mean

Having predicted whether a transaction will occur or not the system needs to predict how much money would be spent. A simple way to make this prediction is taking the mean, however initial testing in MatLab revealed that simply taking an average is affected highly by changes in spending patterns and skewed by outliers. In addition, a spending pattern that suddenly changes (for example one caused by the user changing supermarket) takes too long to be reflected in the prediction.

Supported by the background research on weighted smoothing, the system uses weighted averages to account for this. A weighted average is similar to an average but each value is scaled in it's effect by a weighting factor, this allows the system to give a higher weight to more recent transactions.

The weighted average calculation used is shown in Fig. 3.8 where $w(t)$ is the weighting function for time t, the most recent month is $t = 0$ and $t = n - 1$ is the oldest month.

$$\bar{x} = \frac{\sum\limits_{t=0}^{n-1} w(t) \times x_t}{\sum\limits_{t=0}^{n-1} w(t)}$$

Figure 3.8: Weighted arithmetic mean

3.2.3 Five Model System

Using weighted averages is only part of the solution, the system needs to choose appropriate weights for each transaction and the weights chosen will have a different suitability depending on the spending patterns of the user. During initial research on weighted averages, it was observed that due to the variety of spending patterns caused by users different spending habits, there was not a 'one fits all' solution to weighting. For this reason five different weighting functions were selected and when making a prediction the application selects the weighting algorithm most appropriate for the user.

The five weighting functions were selected from a set of eight (shown in Fig. 3.9) after experimentation with personal finance data in MatLab. They were selected for significant differences in behaviour. There are four main function types: Exponential relationship $w_x = e^x$, decay $w_x = 1/x + 1$, power $w_x = x^1$ and static $w_x = 1$. One exponential, power and static were selected, and two examples of decay with a variable to affecting the speed of the decay. It would be possible to include a scaling parameter (decay constant) for each weighting function, leading to adaptive weights and to use a learning algorithm to select the optimal value for that parameter, this is discussed in Section 7.2.3.

To select the best fit weighting function for each user, the system splits the complete

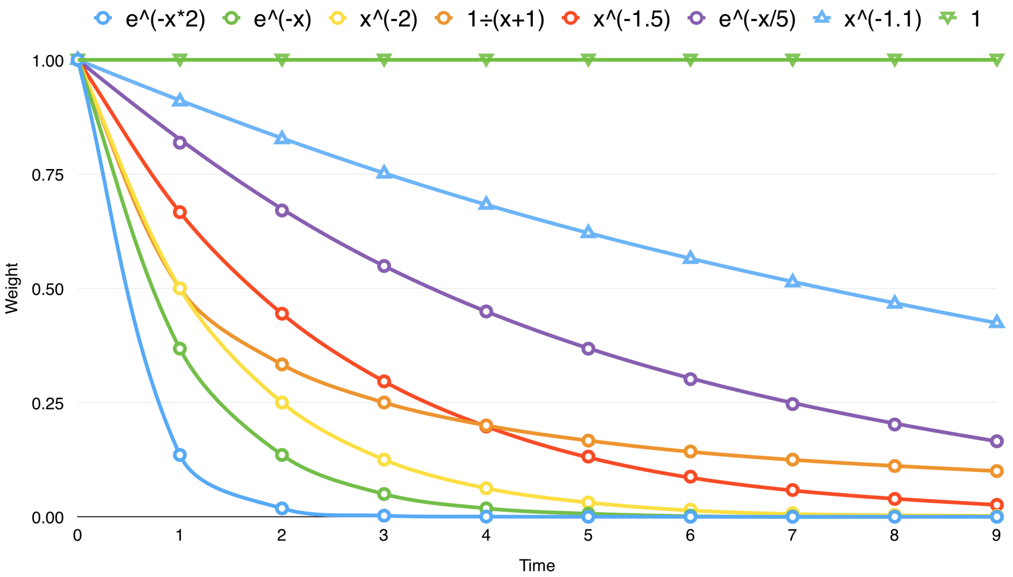


Figure 3.9: The eight prototype weighting functions

months[4] from the users transaction history into two parts. Training contains 75% of months, starting with the oldest and the remaining 25% (the most recent) is used for testing. If less than four months are available the most recent month is used for testing and the remainder for training. In the case where between zero and two months are available the application falls back on a simple average.

Having split the data, the application loops through all the weighting functions available, calculating the weighted average of the testing data and evaluating the mean absolute error (Fig. 3.10) on the training data, by comparing the prediction to the actual value. The function with the least absolute error is best fit to the users overall spending pattern, and so it is selected.

In testing it was discovered that users spending money in similar categories often best suited the same weighting model. Upon further investigation it was discovered that by finding the best weighting model per category in addition to user, on average the absolute error was less. For this reason the most recent implementation of the five model system goes through a users spending in each category, selects the best model for each and uses that to make the prediction in the category. Further research could be done into different levels of modelling and the effect of this on the fit of the model to the user, this is discussed in Section 7.2.1.

3.2.4 Confidence

As the prediction from the Markov Chain Model is based on probabilities it is unstable. To account for this, when making a prediction, the application repeats the process of reading from the MCM up to 10,000 times per second. The repetition of the reading

[4] Months that have passed fully

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$$

Figure 3.10: Mean absolute error where f_i is the prediction and y_i is the true value

is used to produce a confidence level of the final prediction once all the results are combined, which is displayed to the user. The confidence level is displayed as a plus/minus value next to the prediction and gives an indication of how sure the application is.

Assuming the results follow a normal distribution the 95% confidence interval is calculated by Fig. 3.11 where $\bar{x}$ is the arithmetic mean of the predictions, x_i is the value of prediction i and z is a value sampled from a standard normal table for the desired confidence interval.

$$\bar{x} \pm z\frac{\sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \bar{x})^2}}{\sqrt{n}}$$

Figure 3.11: Confidence Interval formula

3.3 Security Considerations

Strong security is expected of this project. The design considers possible attack vectors and takes steps to prevent or reduce the effectiveness of those attacks.

3.3.1 Account Hijacking

To prevent the security concerns highlighted in section 2.3.1 the entire project uses HTTPS, marks cookies as HTTPS only[5], and redirects users to the HTTPS version if they attempt to access via HTTP. This ensures user data is sent encrypted end to end and cannot be intercepted, preventing access to their authentication details or session cookie. In addition cookies are marked as `HttpOnly`, ensuring access via non-HTTPS methods such as client side javascript is not possible. This means that even if a users browser is infected with a malicious script for example using XSS (see 3.3.4), the contents of the cookie cannot be read.

[5] Using the Secure attribute

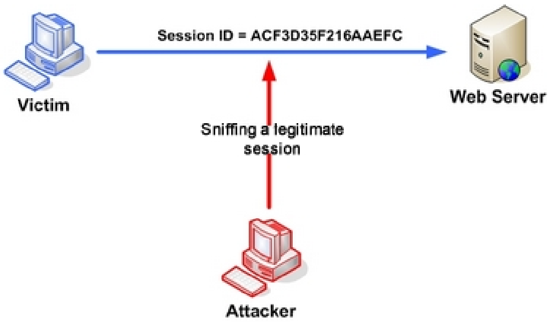


Figure 3.12: Obtaining a users cookie using a MitM attack or Sniffing [23]

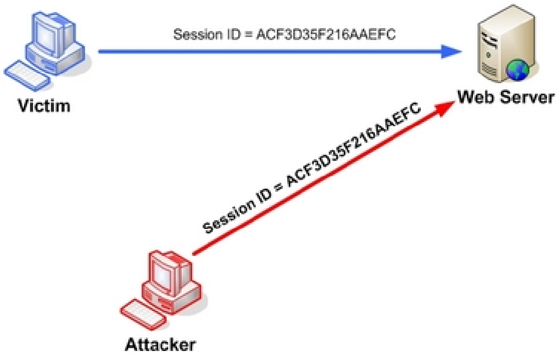


Figure 3.13: Performing a session hijack using another users cookie [23]

3.3.2 Password Security

The project uses Shannon's original equation, using a formula to calculate the probability of guessing each individual character. The formula takes into account that using a larger character set (such as numbers and symbols) decreases the likely hood of successfully predicting the following character and if it falls below a predefined entropy rejecting the password. Enforcing an entropy threshold rather than enforcing a set of restricting 'password rules', was preferred as it gives the user more flexibility hopefully avoiding the annoyance of rules and increases the search space of the passwords. A very long alphanumeric password such as 'correct horse battery staple' would be just as valid as a short password containing numbers and symbols such as '6?@7a?Y5R='.

As part of a brute force attack, the attacker may use a dictionary of popular passwords to reduce the testing space before attempting an exhaustion attack. In order to reduce the effectiveness of this kind of attack the project tests any user provided password against a dictionary of at least 50,000 common passwords sourced from password cracking resources [28].

In addition, to limit the overall effectiveness of brute force attacks, the website rate limits login attempts. If a user attempts to login more than 5 times within one minute, they must wait thirty seconds before they are able to attempt to login again. Rate limit-ing was chosen over CAPCHA[6] found on many websites as CAPCHA's slow down users, are often illegible and visual CAPCHA's can prevent visually impaired users from accessing the website [36], [37]. Additionally CAPCHA's can now be solved automatically with a very high success rate using computer vision techniques, and these techniques are already being integrated into brute force software available online [38]–[41].

3.3.3 Database Storage

As detailed in section 2.3.3, security considerations of storing information in a database were considered. Three main techniques for ensuring the security of the users information stored in the database were employed.

Passwords

Passwords are hashed and salted and different hashing functions were investigated. Traditionally functions such as MD5, SHA1 and SHA256 are used to perform the hashing, however due to advances in modern computer equipment it is possible to generate these at an incredibly fast rate, reducing the time taken to brute force a hash. Using a deliberately slow hashing function is designed avoid this problem. Blowfish written by Bruce Schneier is commonly suggested, as it is designed as a computationally expensive operation [42] . This was evaluated with a simple test, counting the number of hashes completed in one second on the server hosting the project. Table 3.3, shows the results, which found that, on average, Blowfish took significantly longer to generate each hash[7]. For this reason the project salts all passwords and hashes them using Blowfish.

	Hashes Per Second		
	Average	Standard Deviation	95% Confidence Interval
MD5	2,296,667	12,923	±8010
SHA1	1,869,725	14,783	±9162
BLOWFISH	17	0	±0

Table 3.3: Comparison of hashing algorithms hash rate on a 2.7Ghz i7

Personally Identifiable Data

Another concern is personally identifiable data being leaked. In an attempt to avoid this the application encrypts all information stored in the user table, that is needed at a later date using the AES128 encryption standard. This standard was selected for the project as was endorsed by the U.S. National Institute of Standards and Technology, when outlined by NIST in 2001 and has become the "encryption standard for commercial transactions in the private sector" [43], [44].

[6] Completely Automated Public Turing test to tell Computers and Humans Apart

[7] The code used to perform the test can be found in Appendix B

Hashing of Usernames

In addition to the encrypting of data needed at a later date, the username of each user is hashed so it is only known to the person using that account. In the rare case that any of the passwords were brute forced, the relevant username would also need to be brute forced in order to attempt a login or use the details on another website.

3.3.4 Other

Other attack vectors including SQL injection[8] and cross-site scripting[9] (XSS) were also considered.

It was decided that the project would use a prepared statements to reduce the risk of SQL injection. By sending the query followed by the parameters as literal values the database server would not interpret them as an executable portion of SQL and attacks, relying on escaping SQL such as '␣OR␣1 are prevented.

In order to mitigate the possibility of XSS the project will need to escape all content before displaying it to the user or saving to the database. It was decided that the project would use a templating language that escaped output by default, requiring the output be explicitly marked to avoid escaping. By escaping all content before displaying it to the user a maliciously crafted piece of text such as <**script**>alert(1);</**script**> would be sent to the users browser as &lt;script&gt;alert(1);&lt;/script&gt; and not interpreted as a script.

3.4 Technical Design

It was decided that the project would be implemented as a web application. Use of a web application ensures the features of the project can be accessed from anywhere with Internet connectivity and that it is not tied to a particular piece of hardware or operating system.

3.5 Language Choice

As the project is a web application, the user interface was written in HTML & CSS, with the interactive elements in JavaScript. PHP was used on the service side to process the user requests, render the page and access the database which was implemented using MySQL.

PHP was selected because the author had extensive experience in the language, it's original intended use was web development and it is well supported. It's intention of web development means it's well coupled with HTML and provides many libraries to handle it, giving the language a significant advantage over languages such as Python and Java which can be adapted for web use. A key hindrance of PHP is that is it

[8] A code injection technique that uses maliciously crafted statements to modify the SQL executed

[9] Inserting client-side scripts to a websites HTML to customise it's logic

weakly typed which lead to issues during development where objects were passed into functions expecting another object type, weakly typed languages, such as PHP, have no compile-time validation to identity this mistake.

MySQL was selected as the database language because it's well supported by PHP and because it provides Natural Language Searching out of the box, which supports free-text queries and can calculate the relevance of a record in the database to a search term. This is used to power the suggestions feature of the website and for user entered searches.

3.5.1 Design Patterns

The project is implemented using object oriented programming (OOP). OOP is an example of a modular programing, where 'Objects' have attributes and methods. In PHP objects are instances of classes, and these classes interact with each other to perform a task.

During both the design and implementation phases the General Responsibility Assignment Software Pattern (GRASP) design principles outlined by Larman were followed when assigning responsibility to objects within the application. These principles were designed to encapsulate well tested principles of object-oriented design used to identify 'good' software [45].

High Cohesion

Cohesion is a measure of how focused the responsibility of an object is. Every object should have a highly focus purpose and only perform tasks associated with that focus. Ensuring high cohesion throughout the project results in classes are easier to understand, maintain, reuse and adapt as modification are unlikely to affect other parts of the system. High cohesion also lends itself to low coupling, another principle, which assigns responsibility to ensure low dependency between classes, that is a particular component has as little knowledge about other components as possible.

All functionality related to a particular topic has been split into separate classes and in some cases, such as the prediction system, several subclasses.

Controller

The application follows a model-view-controller pattern (MVC), which assigns responsibility of dealing with service events to a controller, encapsulates the business logic in the model and handles the user interface separately in the view. The controller is responsible for receiving the raw web requests, routing them to appropriate part of the application and then returning a rendered page to the user to be viewed in their web browser. Use of the MVC pattern is also an example of the indirection pattern, used to reduce coupling, which uses an intermediate object (the controller) as the interface between two other systems (the view and the model).

In this project, the model was split into several subsections (by functionality) to encapsulate the logic even further and these sub-models handle and manipulate the shared objects in the database. The responsibility of the objects and modules was assigned following the information expert principle, objects which hold the information necessary to complete a task, should be responsible for that task. The view is handled by a templating engine that allows the user interface to be defined in template files, rather than application logic. The output of the models is passed to the view, via the controller, which returns a complete HTML page to send to the user.

Protected Variations

The Protected Variation principle states that elements of the design that are likely to change in the future should be protected from this change by being accessed through an interface. The database access system, which is abstracted from the underlying database server, is an example of Protected Variation found in this project.

3.5.2 Architecture

The service layer of the application uses an follows a MVC request pattern, shown in Fig. 3.14. A HTTP request sent to the server is handled by routing, passed to the appropriate part of the controller, which interacts with the model before generating the view and returning it to the client.

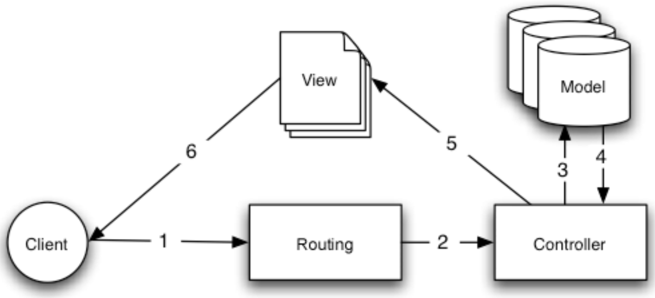


Figure 3.14: The MVC Request architecture of the application's service layer [46]

The Transaction object, shown in Fig. 3.15, which represents individual transactions on a users statement is the core of the applications model. It is coupled to the User, Import, Category and Transactor objects. The Import object is associated with an upload of a users statement, and contains a summary of the uploaded files contents including the date range that was found within the file. Transactions are (indirectly) associated with a Transactor, which represents a real world business that money can be sent to or from.

A full database schema for the application is shown in C on page 85, further examples of system architecture are shown in section 4.2.3 and section 4.3.

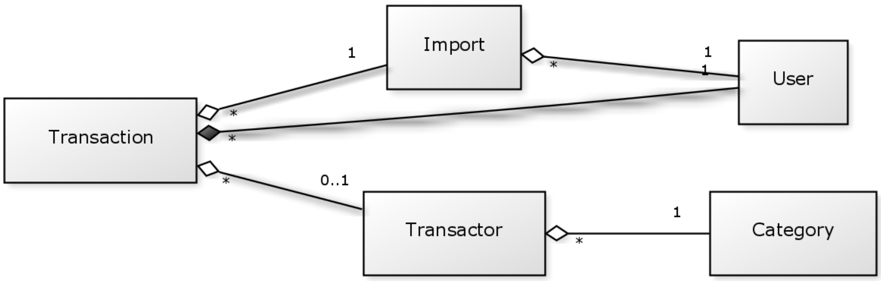
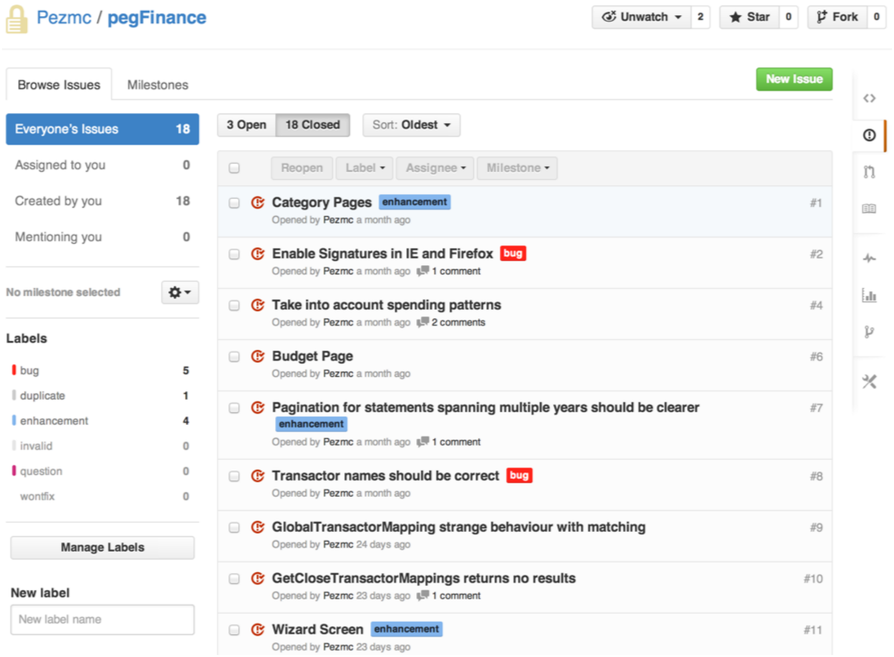


Figure 3.15: Classes coupled with the Transaction object

3.5.3 Project Management

The project was developed in an iterative manner. Key pieces of functionality identified in the planning process were designed, implemented, deployed and tested first. The functionality was then reviewed by potential users, the developer and the project supervisor using lab observations and face-to-face conversations. The results of these discussions were used to decide on the next piece of functionality to implement and to adjust the plan as appropriate, restarting the process. Using an iterative method ensured that the project was build incrementally, starting with the most important features first and that a copy of the application, with the completed stable features was always available for testing and evaluation.

The project code base was managed using the distributed version control system Git. GitHub was selected as the host for the Git repository to keep an external backup of the code base and to access it's project management features [47]. Functionality and bug fix requests created during the testing and review phases were managed using the issues system (Fig. 3.16). To start a new iteration, a milestone, containing a subset of the issues listed select by priority, was created, and this milestone was associated with a new git branch. Once the features were complete, the branch merged into the master branch, which contained a stable copy of the project at all times, and automatically deployed to the web server using post commit hooks. After completing the merge, the milestone, and all associated issues were marked complete and the process was repeated.



Pezmc / pegFinance

Everyone's Issues	18
Assigned to you	0
Created by you	18
Mentioning you	0

No milestone selected

Labels

bug	5
duplicate	1
enhancement	4
invalid	0
question	0
wontfix	0

- Category Pages enhancement — Opened by Pezmc a month ago — #1
- Enable Signatures in IE and Firefox bug — Opened by Pezmc a month ago 1 comment — #2
- Take into account spending patterns — Opened by Pezmc a month ago 2 comments — #4
- Budget Page — Opened by Pezmc a month ago — #6
- Pagination for statements spanning multiple years should be clearer enhancement — Opened by Pezmc a month ago 1 comment — #7
- Transactor names should be correct bug — Opened by Pezmc a month ago — #8
- GlobalTransactorMapping strange behaviour with matching — Opened by Pezmc 24 days ago — #9
- GetCloseTransactorMappings returns no results — Opened by Pezmc 23 days ago 1 comment — #10
- Wizard Screen enhancement — Opened by Pezmc 23 days ago — #11

Figure 3.16: Enhancement and bug fix requests as issues on GitHub

Chapter 4

Implementation

This section is focused on particularly interesting parts of implementing the design outlined in chapter 3 and use of external libraries.

4.1 Key External Libraries

A full list of external libraries and frameworks used by the project can be found in Appendix D.

4.1.1 Server Side

Framework

The project uses a PHP Framework, written by the report author, to handle functionality that is common across different web applications, including request routing, template generation and caching.

Using the framework has several key advantages over writing the whole system from scratch:

Model View Controller MVC is enforced ensuring ensures models (data structures), views (page content) and controllers are separate, improving cohesion and decreasing coupling

Code Structure Code is structured in a standardised manner, split by functionality and is not executable over the web

Rapid Code Development Functionality such as routing, security and caching, that is common between web applications is already implemented

Vendor Libraries External libraries supported by the framework can be used without additional code, such as Twig, the template engine

```
<?php if($items): ?>
  <?php foreach($items as $item): ?>
    * <?php echo htmlspecialchars(strtoupper($item->name), ENT_QUOTES, '
        UTF-8') ?>
  <?php endforeach; ?>
<?php else: ?>
    No item has been found.
<?php endif; ?>
```

(a) PHP as a Templating Engine

```
{% for item in items %}
  * {{ item.name }}
{% else %}
  No item has been found.
{% endfor %}
```

(b) TWIG as a Templating Engine

Figure 4.1: Comparison of TWIG and PHP for a simple template

Twig

Twig is a PHP template engine that takes collections of PHP objects and following a provided .html template generated the HTML output that is ultimately sent to the browser. Although PHP itself is by design a template engine, using a dedicated templating engine allows separation of concerns and addresses the shortcomings of PHP as a template language.

Twig includes idioms for common needs including a `for/else` structure which loops through a list or, if that list is empty, evaluates the else clause and template inheritance; is unit tested; caches all generated templates leading to significant speed increases for the end user and provides shorthand filters such as [1, 2, 3, 4]| first but most importantly, escaping is enabled by default. In twig all data output as part of a template is converted to the equivalent HTML entity so it cannot be parsed as a script or HTML. A comparison of Twig and PHP for a simple page containing a for loop and output escaping is shown in Fig 4.1.

Twig was selected over alternative engines as it is both well documented and supported, supports extensions out of the box which can be used to add additional functionality and as shown in Table 4.1 is by far the fastest when compared to alternatives [48].

Propel

Propel is an object relational mapping (ORM) library for PHP, which allows the storing and loading objects to and from a relational database. Propel takes, a database schema defined in XML and generates PHP objects that representing the schema which can be saved to, and loaded from, the database [49].

Library	Time (sec)	Memory (Kb)	Renders per second
Twig	3	1,190	3,333
PHPTAL	3.8	2,100	2,632
Dwoo	6.9	1,870	1,449
Smarty 2	12.9	2,350	775
Smarty 3	14.9	3,230	671
Calypso	34.3	620	292
eZ Templates	53	5,850	189

Table 4.1: PHP Templating engines compiling and rendering a simple page 10,000 times [48]

Use of an ORM has four main advantages, it abstracts the database layer providing a common interface protecting against change, it allows for business logic to be encapsulated in the generated objects and not the database, it helps address the cross cutting concerns of database access and wraps common database design patterns in easy to access functions. In addition to these common ORM features, Propel wraps object inheritance in an API, for example to get all the transactions that a user has made \$user−>getTransactions() is called which returns an array of Transaction objects.

4.1.2 Client Side

Bootstrap

Bootstrap is a CSS framework for creating user interfaces. It provides a grid layout system, pre-designed reusable components such as buttons; implements common CSS design patterns and supports responsive web design, as well as gracefully degrading when using older browsers or mobile devices [50].

An extended version of bootstrap is used for the front end of the project, though the colour scheme and layout has been heavily modified to ensure the application doesn't look like a 'Bootstrap website' which is a common complaint of web designers and to ensure a common brand [51], [52].

jQuery

jQuery is a JavaScript development framework that abstracts differences in browser behaviour and provides shorthands to common JavaScript tasks such as AJAX requests, element selection, and HTML traversal and manipulation. It's used throughout the website for javascript user interface effects and powers the core of the suggestion wizard, which is detailed further in subsection 4.2.3. It was chosen for the project over other JavaScript frameworks as it is; extendable, open-source, , provides method chaining and is supported by large respected companies on the web including Google and Microsoft [53].

```
foreach(preg_split("/((\r?\n)|(\r\n?))/", $this->OFXContent) as $line){

          // Trim whitespace
          $line = trim($line);
          if ($line === '') continue;

          // Convert charset to UTF-8
          $line = iconv($charset, 'UTF-8', $line);
          if (substr($line, -1, 1) !== '>') {
            list($tag) = explode('>', $line, 2);
            $line .= '</' . substr($tag, 1) . '>';
          }
          $buffer .= $line ."\n";
}
```

Figure 4.2: Converting SGML to XML

4.2 Statement Management

4.2.1 Upload

OFX is stored an XML like format called Standard Generalized Markup Language (SGML). XML is a subset of SGML that relies on opening and closing tags to idenfity information. PHP has inbuilt libraries for parsing XML such as SimpleXML and using those existing functions was preferred to implementing a custom parser for SGML. As SGML is similar in structure to XML it is possible to convert an SGML file to XML using post processing to modify the tag structure [54], [55].

When parsing the OFX, the application first attempts to parse it as an XML file and captures any exceptions. If exceptions are found it attempts to convert the OFX file to XML and repeats the process. The conversion is performed by considering each line of the SGML, trimming the whitespace on either side, converting the contents to UTF-8 so it's valid in XML and if no closing tag is found, which is a requirement of XML, appending the closing tag. This implementation is shown in Fig. 4.2.

Having parsed the SGML into objects it can just be navigated following the specification and converted to arrays of information for each transaction, referred to as movements. These movements look like XYZ; and can then be converted into the Transaction objects most of the fields are handled automatically but dates need to be handled specifically.

QIF is handled in a different manner as there are no native libraries to parse its format. As per the Fig. 3.2 shown in section 3.1.1 shown in the format is a list of individual movements, with each terminated a .̂ Each field associated with an individual movement is identified with a letter, explaining the contents of that line. The field types that the project was interested in and parses are shown in Table 4.2.

In a similar manner to parsing the OFX files, the application loops though all the lines found in the file, switching on the identifier and sorting the information into an array representing that movement.

D	Date
T	Amount
C	Cleared status
P	Payee
M	Memo
ˆ	End of entry

Table 4.2: QIF fields parsed

```php
$id = substr($line, 0, 1);
$content = trim(substr($line, 1));

switch ($id) {
  case '^': // End of entry
    $this->movements[] = $newMovement;
    $newMovement = array();
    break;

  case 'T':   // Amount
    $newMovement["value"] = floatval(preg_replace("/[^0-9.-]/", "",
       $content));
    break;

  case 'D':   // Date
    $newMovement["date"] = $content;
    break;

  // etc ...
```

Figure 4.3: Parsing QIF transactions using the line identifier

As outlined in section 3.1.1 the conversion from the array to a Transaction object is slightly more involved due to unknown date format. To implement this in PHP, before beginning the conversion, all movements are tested against two regular expressions (regex), representing the possible formats of the dates, which are visualised as graphs in Fig. 4.5 and 4.6. Having considered all the movements found in the file, the application has either decided on one of the date formats or prompts the user to decide the appropriate format, as seen in Fig. 4.7.

4.2.2 Named Entity Resolution

Following the implementation of the mapping architecture outlined in the design, the majority of the named entity resolution is done in the setTransactor method of the Transaction object, which is called when setting the name of the transactor for a each transaction found in the uploaded file. The method performs three key actions, tidying the string, removing common notation added by banking institutions and checking the database for known user and global transactors, otherwise creating a new one. To avoid

```
$dmy = true;
$mdy = true;
foreach($this->getMovements() as $movement) {
  if(!preg_match('#(0[1-9]|[12][0-9]|3[01])[-/](0[1-9]|1[012])
      [-/](19|20|21)?\d\d#', $date))
    $dmy = false;

  if(!preg_match('#(0[1-9]|1[012])[-/](0[1-9]|[12][0-9]|3[01])
      [-/](19|20|21)?\d\d#', $date))
    $mdy = false;
```

Figure 4.4: Identifying the date format using regular expressions

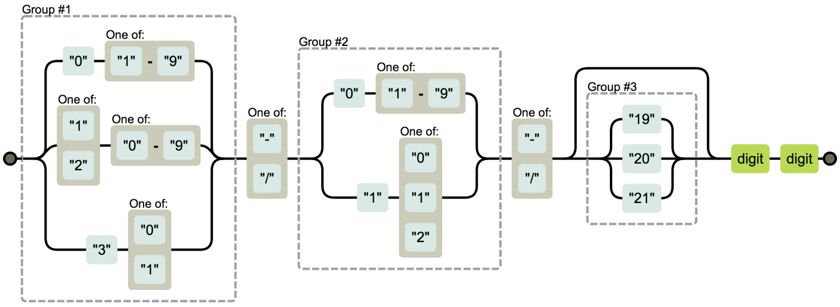


Figure 4.5: Railroad diagram of the regex used to match format d-m-Y

unnecessary duplication in the mappings table the transactor names are normalised before checking for an existing name, this normalisation was added to combat the various different ways which banks store the transactor names which were discovered during testing. The modifications included: padding with spaces, replacing spaces with underscores/tabs, or including non alphanumeric letters; some examples are shown in Table 4.3. Having normalised the input, the next step was to remove any numeric identifiers which had been added and move that detail to another field. Examples of such identifiers included store id's for shops with multiple outlets and account numbers for transfers between accounts, if this detail wasn't removed a separate mapping would be created for transactions referencing the same entity, leading to unnecessary duplication. The identifiers were removed using a regex, expressed in Fig. 4.8 which only captures numbers found at the end of the string requiring at least two digits, these additional constraints were added to preserve transactors with numbers in their name, such as `H3G` and `SUPERMERCADO␣3`. The final step checks for existing GlobalMapping or UserMapping objects in the database and if found associates that mapping with this transaction. If neither mapping's are found a new UserMapping object is created, persisted and associated with the transaction. Differing from the original plan, fuzzy matching[1] of the transactor name when searching for existing transactors is not performed as this functionality was moved to the suggestion wizard. The full code for the 'setTransactor' method is shown in Fig. E.1.

[1] Finding strings that approximately rather than exactly match

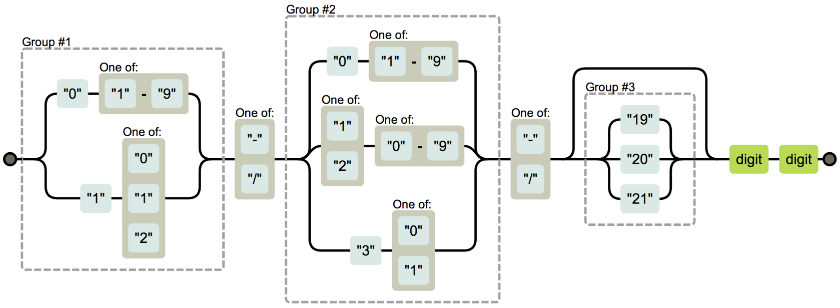
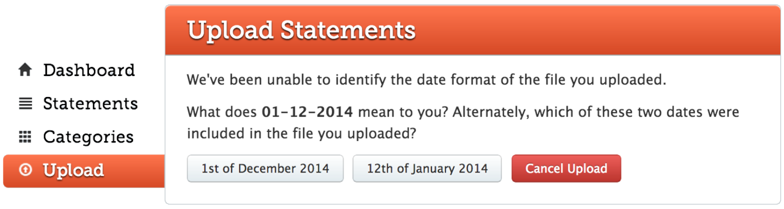


Figure 4.6: Railroad diagram of the regex used to match format m-d-Y



Upload Statements

We've been unable to identify the date format of the file you uploaded.

What does **01-12-2014** mean to you? Alternately, which of these two dates were included in the file you uploaded?

Figure 4.7: UI Prompt when the application is unable to decide an upload's date format

4.2.3 Suggestion Wizard

The suggestion wizard (Fig 4.9) was added to the project in response to observations during lab usability testing. Originally the suggestions were only found on each unmapped transactors page (Fig. 4.11) to speed up the process of mapping a reference to an existing transactor. Users appeared to spend a lot of time manually categorising each individual transactor by searching their statement for an unknown transactor, opening that transactors details and then filling in the appropriate forms or clicking on a suggestion, following an activity diagram similar to Fig. 4.10.

The wizard was designed to streamline and speed up this process as well as making it easier for the user to follow. Each unmapped transactor is displayed in turn, starting with the one with the most transactions (encouraging the user to map the ones they shop at more often first). On each step, the user can; follow the suggestions, manually map the reference to an existing transactor, create a new transactor or ignore the reference for now. A step-by-step view of this process is shown in section 5.2.

During the process feedback is displayed to the user using notifications that appear over the wizard and are coloured according to validation style convention, with red warning of an error and green confirming success (Fig. 4.12). The notifications themselves were implemented using the PNofity javascript library for jQuery which provides an API for managing alerts send to the user. PNotify was chosen as it had native support for bootstrap styles, which were already being used for the projects UI, and it was open source licensed under the GPL[3] [57].

By following the wizard users are able to quickly map the majority of their transac-

[3] GNU General Public License [56]

Raw Transactor Name	Occurrences
`TESCO␣STORES␣5128`	87
`TESCO␣STORES␣2977`	68
`TESCO_STORES`	14
`SACAT␣MARKS␣␣␣␣AND`	33
`SACAT␣MARKS␣␣AND`	16
`WILKINSON␣`	22
`WILKINSON`	8
`TO␣A/C␣00000000`[2]	2

Table 4.3: Examples of the raw transactor names found on uploaded statements

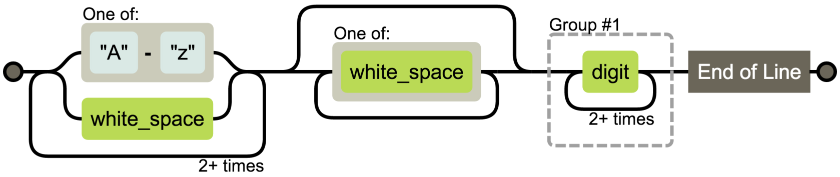


Figure 4.8: Railroad diagram of the regex used to tidy the transactor name

tions in an easy to follow way, as can be shown by the activity diagram in Fig. 4.13.

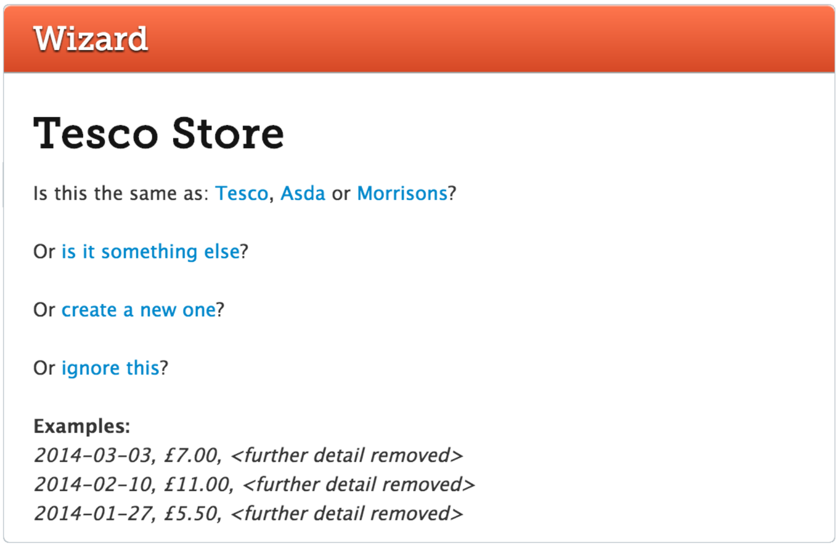
Implementation

The wizard is powered by Ajax[4], which sends requests to a backend RESTful API which proves access to the unmapped transactors found for the user. The requests and responses are sent as JSON[5] which is supported natively by both PHP and JavaScript. A GET request to `/ajax/transactor/suggestions` returns a collection of unmapped transactors including associated examples which are then added to the UI using JavaScript. Mapping (including following a suggestion) and creating a new transactor are both handled by POST requests to the same API. Mapping sends a request to `/ajax/transactor/map` containing the ID of the user mapping and global mapping and creation sends a request to `/ajax/transactor/create` including the user mapping ID to associate with the new transactor. Examples of the JSON communication are shown in F.

On the backend, all JSON serialisation is handled by implementing the `JsonSerializable` interface on all objects that need to be JSON encoded. This allows the native function `json_encode($object)` to correctly encode the object into JSON using the data returned by the abstract method ‘jsonSerialize‘ defined in the interface. The final inheritance diagram for the mapping, transactor and transaction objects is shown in Fig. 4.14.

[4] Asynchronous JavaScript and XML

[5] JavaScript Object Notation



Wizard

Tesco Store

Is this the same as: Tesco, Asda or Morrisons?

Or is it something else?

Or create a new one?

Or ignore this?

Examples:
2014-03-03, £7.00, <further detail removed>
2014-02-10, £11.00, <further detail removed>
2014-01-27, £5.50, <further detail removed>

Figure 4.9: Suggestion wizard UI

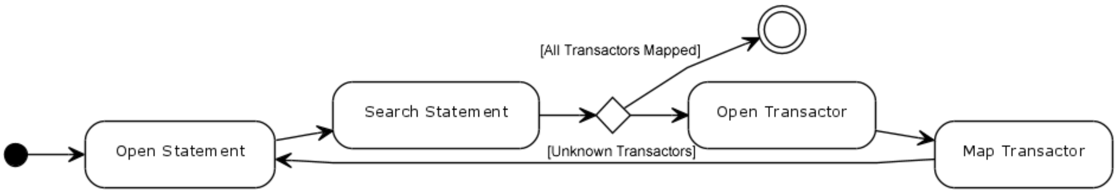


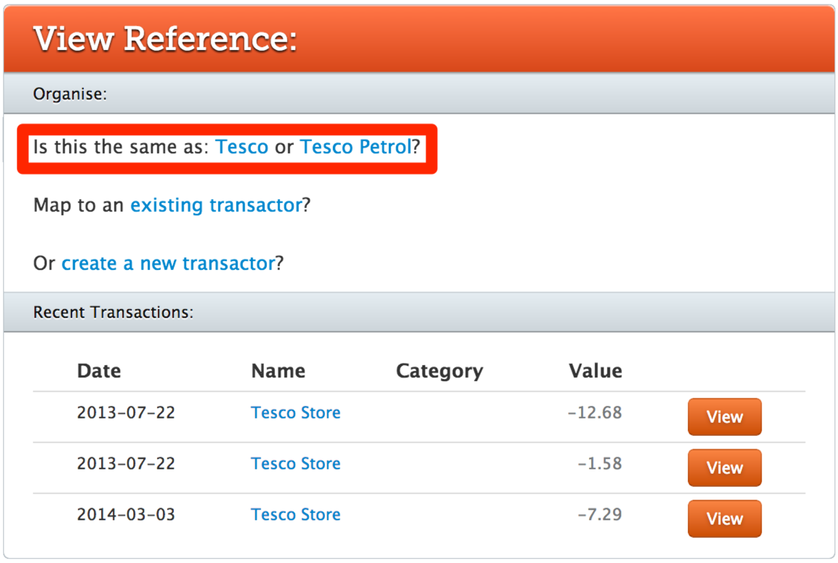
Figure 4.10: Activity diagram for mapping individual transactors

The Suggestions

The suggestions are made by searching for known global mappings, user mappings, global transactors and user transactors[6] that are similar to the reference name a suggestion is being made for. This is done using the Natural Language Full-Text Search functions provided by MySQL. These functions, including `MATCH`, use a Vector Space Model to rank the relevance of a field relative to the input and signify this relevance using a decimal number, which means it can be treated as a standard field by the database and used to order the results. [58].

When searching for suggestions the application finds the five most relevant mappings or transactors, according to full text search, loads those objects from the database using the ORM. As the ORM tool uses doesn't have support for full text search a raw SQL query was written that uses PDO's[7] prepared statements functionality to send the parameters to the database server, avoiding SQL injection. The query for finding global transactor mappings as part of the `GetCloseTransactorMappings` method in the GlobalTransactorMappingPeer class is shown in Fig. 4.15, where `:identifier` is replaced by the

[6] In the order of preference
[7] PHP Data Objects extension for accessing databases



View Reference:

Organise:

Is this the same as: Tesco or Tesco Petrol?

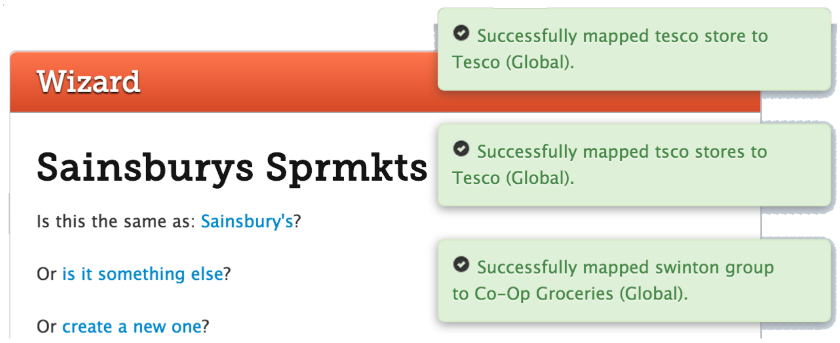
Map to an existing transactor?

Or create a new transactor?

Recent Transactions:

Date	Name	Category	Value	
2013-07-22	Tesco Store		-12.68	View
2013-07-22	Tesco Store		-1.58	View
2014-03-03	Tesco Store		-7.29	View

Figure 4.11: Suggestions shown on the Transactor (reference) page



Wizard

Sainsburys Sprmkts

Is this the same as: Sainsbury's?

Or is it something else?

Or create a new one?

Successfully mapped tesco store to Tesco (Global).

Successfully mapped tsco stores to Tesco (Global).

Successfully mapped swinton group to Co-Op Groceries (Global).

Figure 4.12: Notifications shown during the suggestion wizard

server with the passed parameters.

4.3 Prediction

The key difficulty of implementing the prediction functionality was designing a data structure that allowed predictions to be read by column or by row in order to generate the table output shown in Fig. 4.16. Ordinarily a multidimensional array or a dictionary of dictionaries would be used, however in this case it must be possible to access the data by subcategory as well as by month or date. In order to allow accessing the data by month, category and subcategory as well as being able to generate totals the data structure shown in Fig. 4.17 was used. The TransactionCollection interface defines three methods, `getTotalValue()`, `getTransactions()` and `addTransaction(Transaction)`. By extending this interface each level of the data structure can be treated in exactly the same way and the

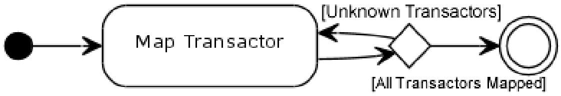


Figure 4.13: Mapping transactors using the suggestion wizard

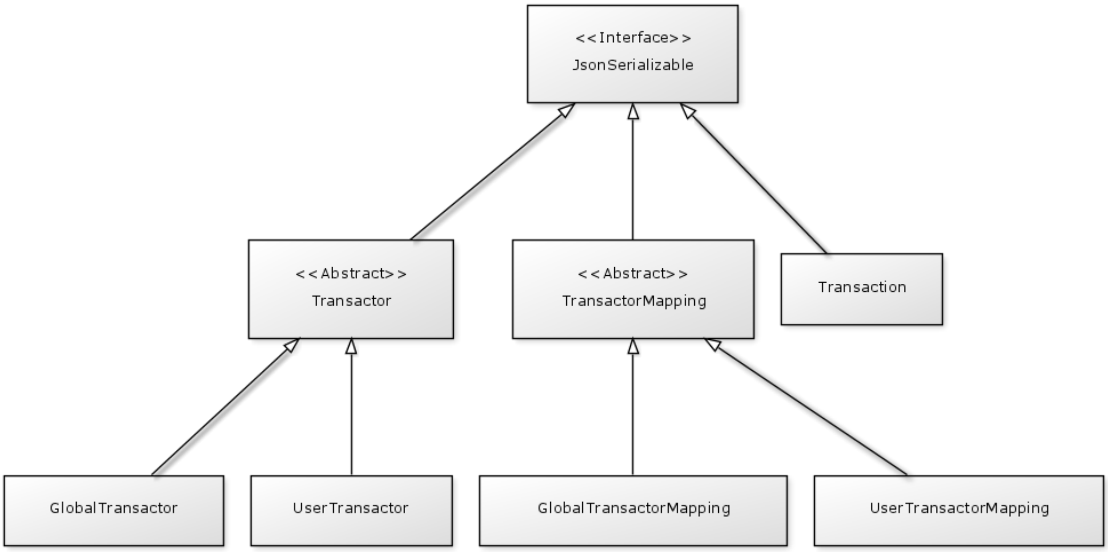


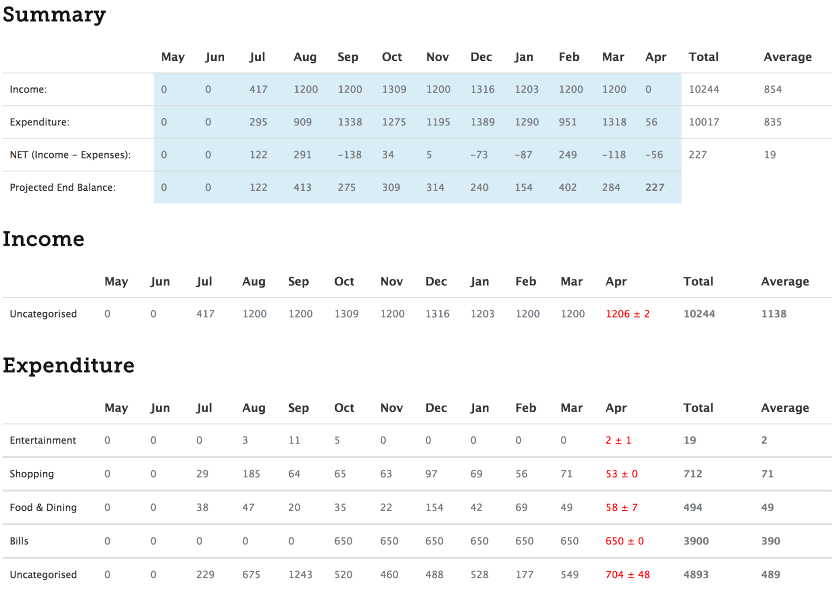
Figure 4.14: Inheritance diagram of JsonSerializable objects

entire PersonalBudget object can be passed to the template engine with no controller logic or pre-calculation. Transactions are added to the PersonalBudget object and at each layer of the data structure organised into the correct collection automatically. When getting the transactions from any collection, the object calls the get `getTransactions()` method on any sub-collection instances it holds and returns an intersection of the results.

The Budget object (representing expenditure or income) provides the `getCategoryPrediction()` method which takes a month and a category and returns the systems prediction for that month. It's this method that builds and samples from the Markov Chain Models, calculates the Weighted Arithmetic Mean, and chooses the Weighting Model which provides the least absolute error for the historical data in order to make the prediction. The class also calculates the confidence interval which is displayed in the UI.

```
SELECT *, MATCH(:nameColumn) AGAINST (:name) AS 'score'
FROM :tableName
  WHERE :transactorID IS NOT NULL
  HAVING `score` > :minScore OR :nameColumn = :name
ORDER BY `score` DESC
LIMIT 5
```

Figure 4.15: SQL query selecting similar mappings using PDO



Summary

	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Income:	0	0	417	1200	1200	1309	1200	1316	1203	1200	1200	0	10244	854
Expenditure:	0	0	295	909	1338	1275	1195	1389	1290	951	1318	56	10017	835
NET (Income – Expenses):	0	0	122	291	−138	34	5	−73	−87	249	−118	−56	227	19
Projected End Balance:	0	0	122	413	275	309	314	240	154	402	284	**227**		

Income

	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Uncategorised	0	0	417	1200	1200	1309	1200	1316	1203	1200	1200	1206 ± 2	10244	1138

Expenditure

	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Entertainment	0	0	0	3	11	5	0	0	0	0	0	2 ± 1	19	2
Shopping	0	0	29	185	64	65	63	97	69	56	71	53 ± 0	712	71
Food & Dining	0	0	38	47	20	35	22	154	42	69	49	58 ± 7	494	49
Bills	0	0	0	0	0	650	650	650	650	650	650	650 ± 0	3900	390
Uncategorised	0	0	229	675	1243	520	460	488	528	177	549	704 ± 48	4893	489

Figure 4.16: The budget overview screen where predictions are highlighted in red

The logic for these steps is split into separate classes to improve cohesion and the overall process is shown in Fig 4.18.

In order to build the Markov Chain Model, an instance of, the TransactionMarkovChain class takes a list of Transaction objects and counts the months where a transaction occurs, storing this in an associative array. This matrix is read to generate a transition table which represents the amount of time a transaction as transitioned from occurring to not occurring and visa-versa. This transition table can be converted to a transition matrix which represents the probability of such a transition occurring. An example transition table and matrix for a one of purchase are shown in Table 4.5 and 4.4. Considering whether or not a transaction occurred in the previous month, a sample is taken from the matrix. For example, if example event occurred last month for the probability of the event occurring next month given that it occurred last month is 0.00 and so there is a 0% chance the system will predict the event will occur next month. This process of sampling is repeated up to $10,000$ times for each transaction in under one second and the list of predictions is returned to the Budget class.

#	Didn't Occur	Occurred
Didn't Occur	10	1
Occurred	1	0

Table 4.4: Transition table for a one off purchase

Having decided whether or not a transaction will occur the Budget class uses an instance of `TransactionWeightedAverageCalculator` to estimate how much money would be transferred if a transaction occurs. The weighted average calculator takes a list of Trans-

#	Didn't Occur	Occurred
Didn't Occur	0.91	0.09
Occurred	1.00	0.00

Table 4.5: Transition matrix for a one off purchase

action objects and returns an associative array containing the weighted average for each transactor. To calculate the weighted averages the calculator first selects the best weighting model for the users spending pattern in the specified category and uses the selected function to calculated the averages for each transactor in the category.

Combining the output of the MCM's and the weighted average calculator the produces the systems expenditure prediction for the considered category, Table 4.6. This is calculated for each sample taken from the MCM and the results are passed to an instance of the PredictionEvaluator to produce an average and confidence interval, which is ultimately displayed on the UI.

Transactor	Prediction	Weighted Average	Total
Tesco	No	£33	£0
Sainsbury's	Yes	£65	£65
Amazon	No	£3	£0
Argos	No	£17	£0
		Total	**£65**

Table 4.6: Combining samples from the MCM and the weighted averages

4.4 Security Considerations

The security considerations outlined in section 3.3, were implemented throughout the project including site-wide SSL, password entropy requirements, password salting and hashing and personal identifiable information encryption. In addition the application implements additional protection against account hijacking and login throttling to reduce the overall effectiveness of brute force attacks.

An overview of the techniques used during login and on every page request is shown in Fig. 4.19 and 4.20.

4.4.1 Account Hijacking

The users session[8] identified by the session ID in the users cookie, includes a fingerprint of the users browser user agent, an additional user session identifier, the last time they performed an action as well as the users ID. These identifiers are validated on every page load and if incorrect the session is invalidated, to prevent further use, and the user is logged out.

[8] Stored server side

The fingerprint and session identifier provide an additional layer of protection against account hijacking and recording the last action time means that if a user leaves their computer idle or forgets to logout their session cannot be used after a certain amount of time has passed.

The user session identifier is generated each time a user logs in, and stored in their user record. On each page load the recorded identifier is compared to the current one found in the users session, if they don't match the user must have an old session and their session is no longer valid. This prevents use of an old session if the user has logged in elsewhere, such as logging in from another computer or mobile device.

Fingerprinting relies on the user agent string (UAS) which sent in the HTTP headers to the remote server by the users browser and contains information including their current browser version and operating system, example UAS' are shown in Fig. 4.21. It's important to note that although the string can be manipulated using a browser extension or by modifying the string sent in the HTTP header the average user would not be doing this and for the purposes of fingerprinting the actual value is not of particular importance. As long as the fingerprint doesn't change value between requests, implying the user is logging in from a different device using the same session (as would be seen during account hijacking), the user is kept logged in.

The use of browser fingerprinting to improve session security was suggested and evaluated by Unger, Mulazzani, Fruhwirt, *et al.* who concluded that the technique was successful in preventing various attacks and hijacking attempts, including XSS and passive sniffing . The paper also demonstrated that FireSheep and WhatsApp Sniffer, discussed in sestion 2.3.1 are thwarted by fingerprinting [59].

The actual fingerprint taken by the project is similar to that suggested in the paper. Consisting of all of the alphabetic strings found in the UAS concatenated and hashed. Only alphabetic strings are used as the likely-hood of a user updating their browser or operating system is high, particularly considering the automatic update feature which is common in many modern browsers [60], [61]. The code used to build the browsers fingerprint is shown in Fig. 4.22.

If a users browser fingerprint changes, their user session ID doesn't match the one in the database, or the time since their last action is over the threshold, their session is both unset[9] and destroyed[10] and their cookie which was used to identify the session to the server is written over[11]. This will cause them to be logged out and redirected to the login page. For convenience the page they were previously accessing is stored and they will be returned to it after successfully logging in.

Destroying the session in this way means that even if an attacker had successfully performed a session hijack before the destruction occurred the cookie that was stolen is no longer valid and cannot be used for authentication.

[9] Frees all session variables [62]

[10] All server side data associated with the session is deleted [63]

[11] Required to 'kill the session altogether' [63]

4.4.2 Brute Force Attacks

Login throttling was implemented using a database table containing every failed login, the time of the login attempt and for reference the IP address and user the failed login attempt was for.

If a user enters an incorrect username or password an entry is added to the failed login attempt table and the application checks up how many failed password attempts have occurred in the last threshold period[12], see Fig E.3. If the number of attempts is larger than the maximum failed logins per period the user must wait for a few seconds before they can try to login again, that is, all login attempts will fail and the UI will prevent login to avoid users becoming confused.

By enforcing a wait time of only a few seconds it's likely a normal user would not notice the throttling as they would take a few seconds to login, but an attack would be seriously hampered. The throttling is applied across all user accounts and IP addresses to ensure use of a botnet to send a distributed brute force attack is also very impractical [64].

In the future the login throttling system could be extended to take into account the average number of logins per day and automatically adjust the throttling thresholds appropriately.

[12] Defined in a config file to allow easy modification

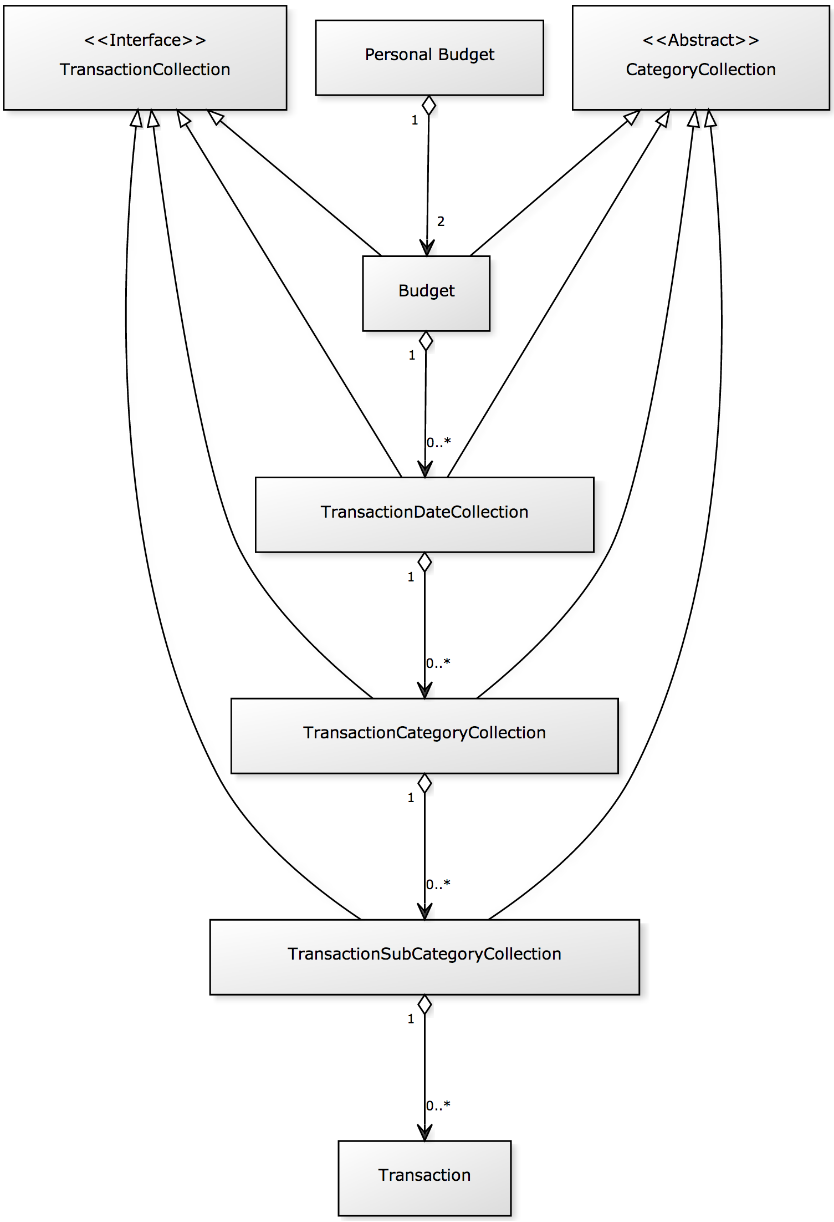


Figure 4.17: Storing a Personal Budget by date and category

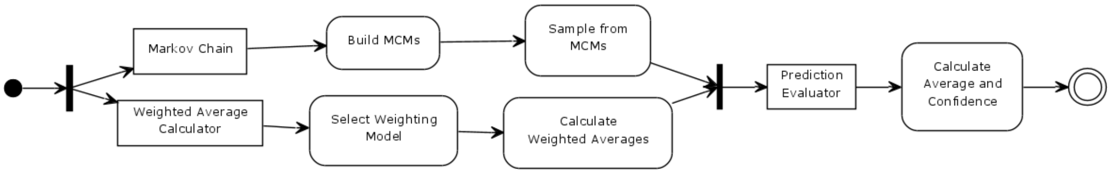

Markov Chain
Build MCMs
Sample from MCMs
Weighted Average Calculator
Select Weighting Model
Calculate Weighted Averages
Prediction Evaluator
Calculate Average and Confidence

Figure 4.18: System activity diagram when making a prediction

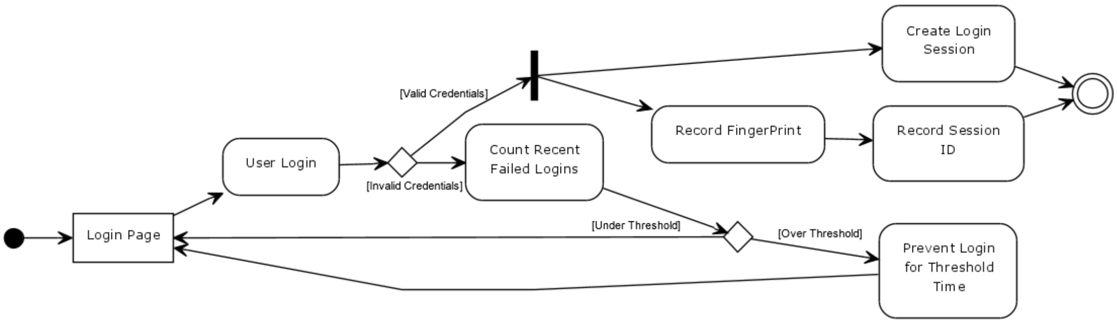


Figure 4.19: Steps performed during a user login

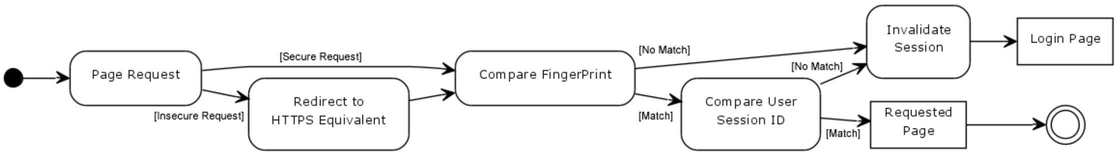


Figure 4.20: Steps performed during every page load request

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/34.0.1847.131 Safari/537.36
```

(a) Google Chrome running on OS X

```
Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X; en-us) AppleWebKit/534.46 (KHTML, like
Gecko) Version/5.1 Mobile/9B176 Safari/7534.48.3
```

(b) Google Chrome running on OS X

```
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
```

(c) Internet Explorer 10 on Windows 7

```
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:24.0) Gecko/20100101 Firefox/24.0
```

(d) Firefox 24 on Linux (Ubuntu)

Figure 4.21: Example User Agent String's

```php
$parts = preg_split('#[^A-z]+#', $_SERVER['HTTP_USER_AGENT']);
$fingerprint = "";
foreach($parts as $part) {
  if(strlen($part) > 1 || strlen($fingerprint) < 2)
    $fingerprint .= $part;
}
```

Figure 4.22: Generating a browsers fingerprint using the user agent string

Chapter 5

Results

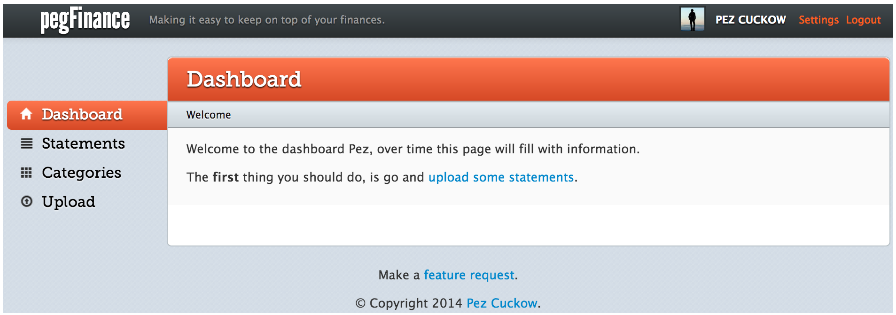
5.1 System Walkthrough

Below, an overview of the systems behaviour is given following the standard users use case in the form of a walk through. For further exploration of features the application can be found online at `https://secure.pezcuckow.com/` where an account can be registered for access.

Having logged in, the application starts on a welcome screen (Fig. 5.1) which fills with information over time, for now the system prompts the user to upload a bank statement. The user bar appears at the top of every page[1] which identifies the currently logged in user by name and a avatar pulled using from Gravatar[65], it provides quick access to the user settings and allows the user to logout.

On the left hand side of the screen is the main menu (Fig. 5.2), which is also displayed on all pages. It indicated the currently open page in orange, in theme with the rest of the website, and on hover with a mouse it highlights the chosen page by inverting the colours to make the selection clear.

[1] For clarity both the background and user bar are not shown in the following screenshots



Welcome to the dashboard Pez, over time this page will fill with information.

The **first** thing you should do, is go and upload some statements.

Make a feature request.

© Copyright 2014 Pez Cuckow.

Figure 5.1: Empty welcome screen

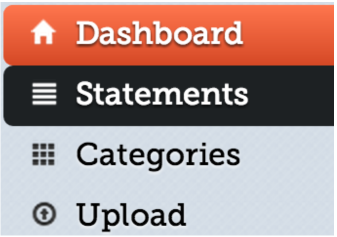
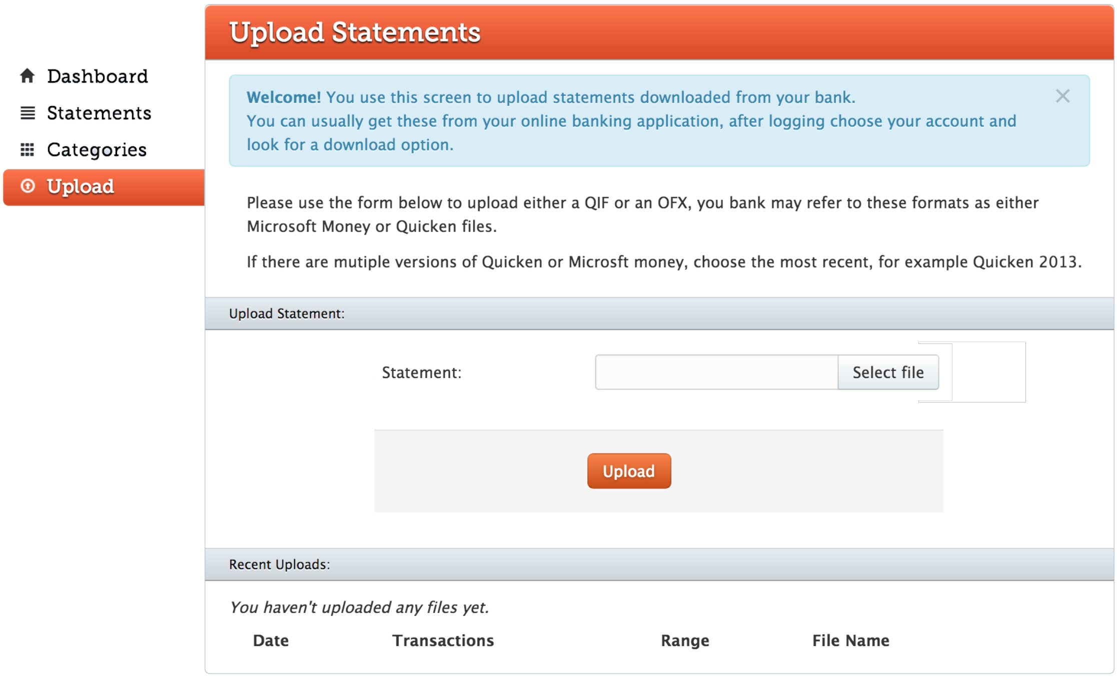


Figure 5.2: Application main menu



Welcome! You use this screen to upload statements downloaded from your bank.
You can usually get these from your online banking application, after logging choose your account and look for a download option.

Please use the form below to upload either a QIF or an OFX, you bank may refer to these formats as either Microsoft Money or Quicken files.

If there are mutiple versions of Quicken or Microsft money, choose the most recent, for example Quicken 2013.

Upload Statement:

Statement:

Recent Uploads:

You haven't uploaded any files yet.

Date	Transactions	Range	File Name

Figure 5.3: The upload statement screen before any uploads

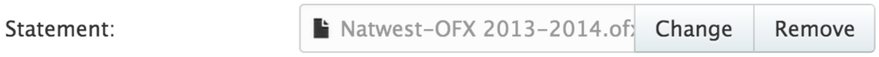
5.1.1 Statement Upload

Following the advice of the welcome page, the user opens the upload screen (Fig. 5.3). As this is the first time user has opened this screen an information prompt is shown which explains the purpose of the page and reminds them that the files to upload can usually be found on their current banks Internet banking system.

After downloading a statement from their account, the user uploads it by clicking on the select file button and browsing their computer for the file. After confirming their selection the state of the file field changes (Fig. 5.4) to allow checking of the uploaded file and to make it clear one is currently selected. Clicking the primary[2] upload button uploads the file to the server, which processes it, and refreshes the page.

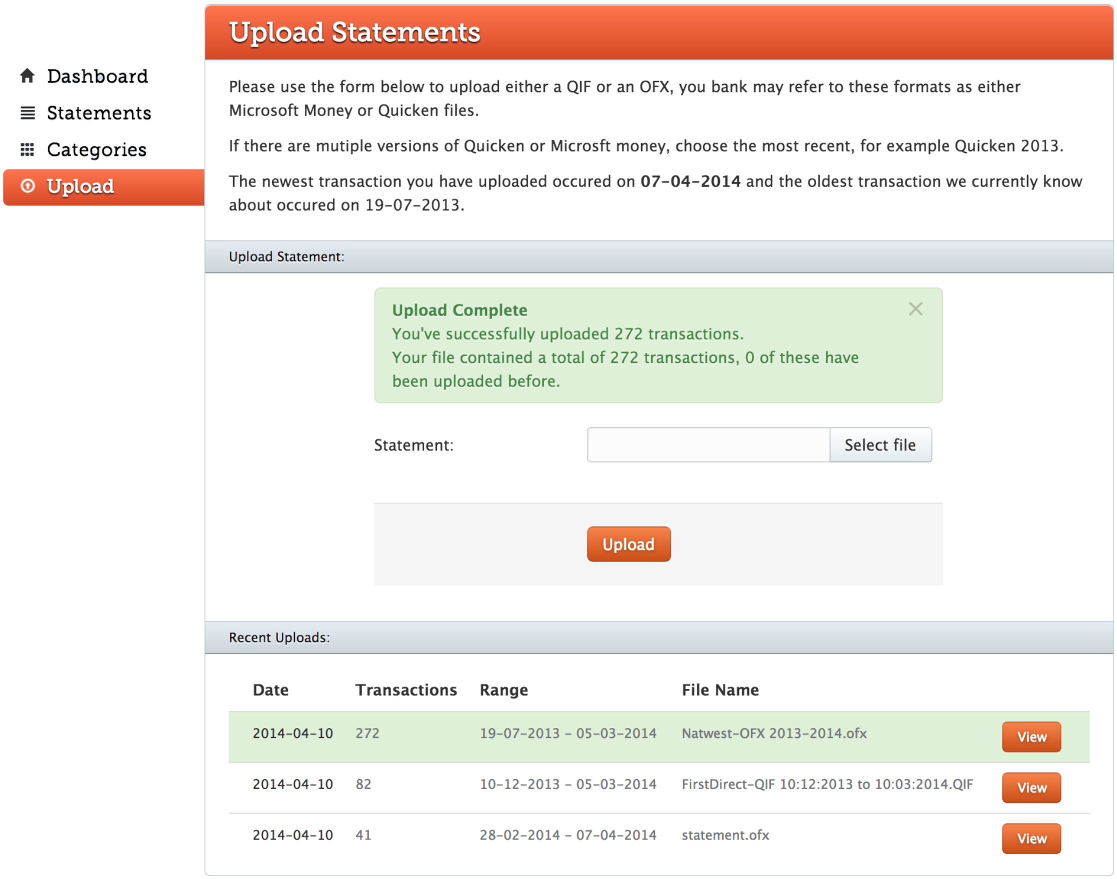
On the new page (Fig. 5.5), the state of the file upload is indicated above the form,

[2] Indicated in orange



Statement: Natwest-OFX 2013-2014.of

Figure 5.4: UI update following a file selection



Upload Statements

Please use the form below to upload either a QIF or an OFX, you bank may refer to these formats as either Microsoft Money or Quicken files.

If there are mutiple versions of Quicken or Microsft money, choose the most recent, for example Quicken 2013.

The newest transaction you have uploaded occured on **07-04-2014** and the oldest transaction we currently know about occured on 19-07-2013.

Upload Statement:

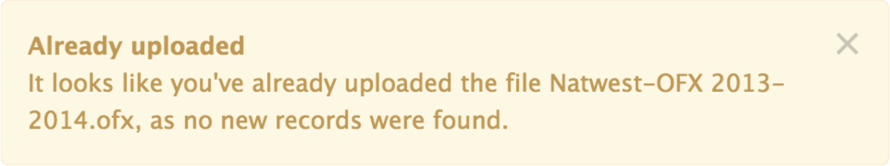
Upload Complete
You've successfully uploaded 272 transactions.
Your file contained a total of 272 transactions, 0 of these have been uploaded before.

Statement:

Recent Uploads:

Date	Transactions	Range	File Name	
2014-04-10	272	19-07-2013 - 05-03-2014	Natwest-OFX 2013-2014.ofx	View
2014-04-10	82	10-12-2013 - 05-03-2014	FirstDirect-QIF 10:12:2013 to 10:03:2014.QIF	View
2014-04-10	41	28-02-2014 - 07-04-2014	statement.ofx	View

Figure 5.5: The upload page, following successful file uploads



Already uploaded
It looks like you've already uploaded the file Natwest-OFX 2013-2014.ofx, as no new records were found.

Figure 5.6: Upload confirmation following a duplicate file

containing a summary of the file uploaded and the uploaded file is highlighted in the recent uploads list, which includes an option to view the transactions uploaded as part of that statement. The highlight colour, green, was used to indicate success.

In addition a further piece of information has been added to the page introduction, the most recent transaction the system knows about is marked in bold, to save the user time when selecting a statement to download.

If the user uploads a file containing no new transactions (all previously uploaded), the confirmation prompt indicates this and it uses the colour yellow to indicate a warning (Fig. 5.6).

On returning to the welcome screen, the dashboard is now full of information[3] (Fig. 5.7). It provides an overview of the average income and outgoings per month, along with an indication of the users 'profit' since the first transaction the application knows about.

Most notably, the dashboard also includes an interactive pie chart, which gives an indication as to which categories most of their money is spent. On hover the chart gives the actual percentage of the overall expenditure, and on click opens a page which lists all the transactions in that category.

If the system has suggestions for uncategorised transactors, a notification suggesting they visit the category (or suggestion) wizard is shown.

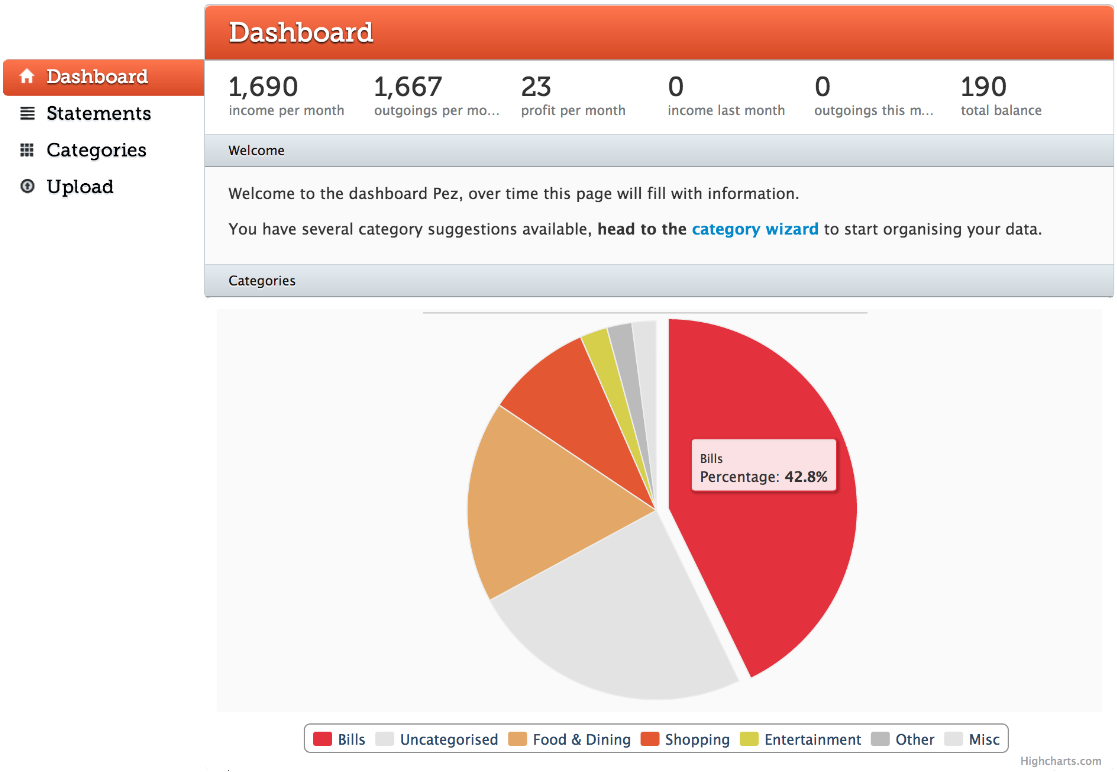
5.2 Suggestion Wizard

The suggestion wizard steps the user though all unmapped and uncategorised references (Fig 5.8). It starts with the transactor they frequent the most often and prompts them to provide the mapping, either by following a suggestion, creating a new transactor or manually searching for a match.

Completing the mapping, forwards the user to the next unmapped reference, providing confirmation the new mapping has been created as a notification which disappears after a few seconds 5.9, and the process repeats.

The suggestions wizard went through several iterations of user interface enhancements, designed to make it easier to use. For example, when creating a new transactor (Fig. 5.10) the system automatically fills in the name field with a pre-formatted version of the reference and the category selection is performed through an upgraded dropdown menu. The dropdown has been upgraded from the standard dropdown found on many

[3] Fictional information used throughout out this walkthrough



1,690 income per month

1,667 outgoings per mo...

23 profit per month

0 income last month

0 outgoings this m...

190 total balance

Welcome

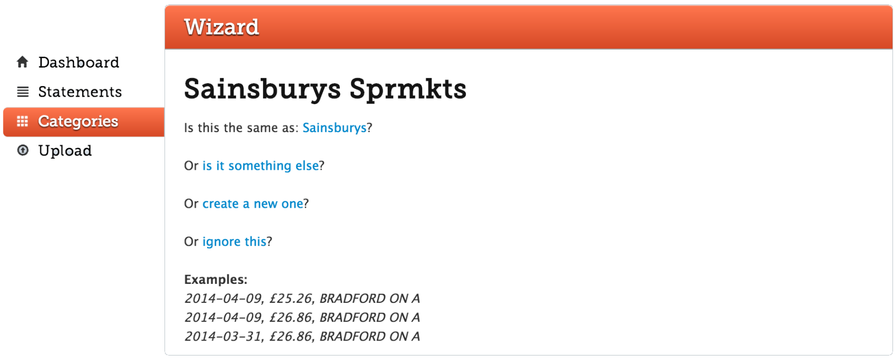
Welcome to the dashboard Pez, over time this page will fill with information.

You have several category suggestions available, **head to the category wizard** to start organising your data.

Categories

Bills
Percentage: **42.8%**

Figure 5.7: Welcome screen following successful statement uploads, including expenditure per category



Sainsburys Sprmkts

Is this the same as: Sainsburys?

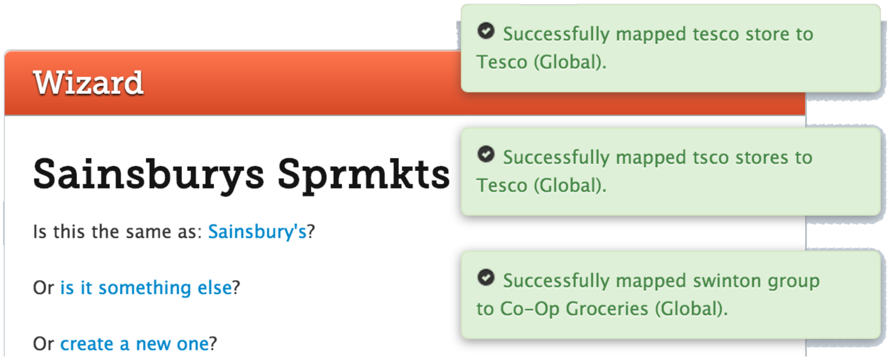
Or is it something else?

Or create a new one?

Or ignore this?

Examples:
2014-04-09, £25.26, BRADFORD ON A
2014-04-09, £26.86, BRADFORD ON A
2014-03-31, £26.86, BRADFORD ON A

Figure 5.8: Suggestion wizard main screen, showing suggestions for a reference



Successfully mapped tesco store to Tesco (Global).

Successfully mapped tsco stores to Tesco (Global).

Successfully mapped swinton group to Co-Op Groceries (Global).

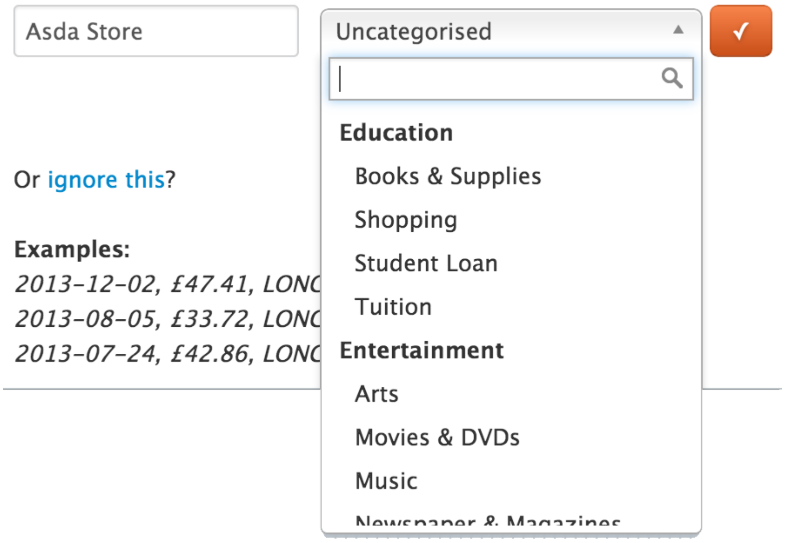
Sainsburys Sprmkts

Is this the same as: Sainsbury's?

Or is it something else?

Or create a new one?

Figure 5.9: Notifications shown after completing a successfully



Asda Store

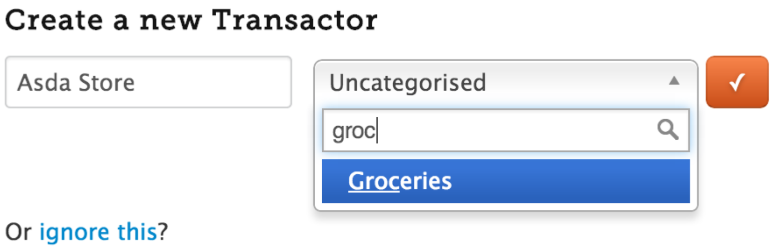
Uncategorised

Or ignore this?

Examples:

2013-12-02, £47.41, LONC
2013-08-05, £33.72, LONC
2013-07-24, £42.86, LONC

Figure 5.10: Creating a new Transactor and selecting a category



Create a new Transactor

Asda Store

Uncategorised

groc

Or ignore this?

Figure 5.11: Selecting a category using the autocomplete feature

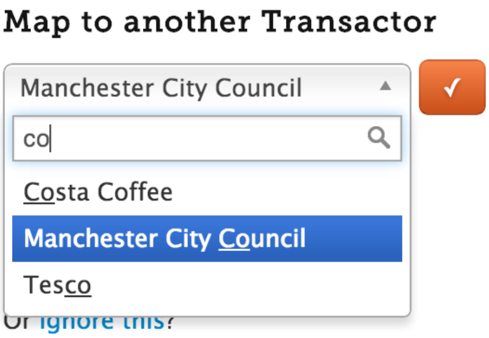


Figure 5.12: Searching for an existing Transactor using autocomplete

websites using client side javascript and provides auto-completion, fuzzy matching, as well as the ability to select a category grouping, for example entertainment over Movies & DVD's (Fig. 5.11. A similar dropdown is also used when searching for an existing transactor (Fig. 5.12.

After completing the wizard by mapping or ignoring all the unmapped references, the user is congratulated and a hint is given that they should head to the transaction summary page, which includes their monthly expenditure predictions.

5.3 Transaction Overview

The transaction overview screen shows spending per month in each of the categories and a the prediction for next month made using the machine learning techniques outlined in section 3.2 (Fig. 5.13). At the top of the screen a summary of the total income, expenditure and net profit for each month is shown. The rest of the page is split into two sections, representing money coming into and leaving the users account. In the example shown the user hasn't mapped all of their references and so some transactions are listed under uncategorised, a hint is placed at the top of the screen reminder users to visit the category wizard.

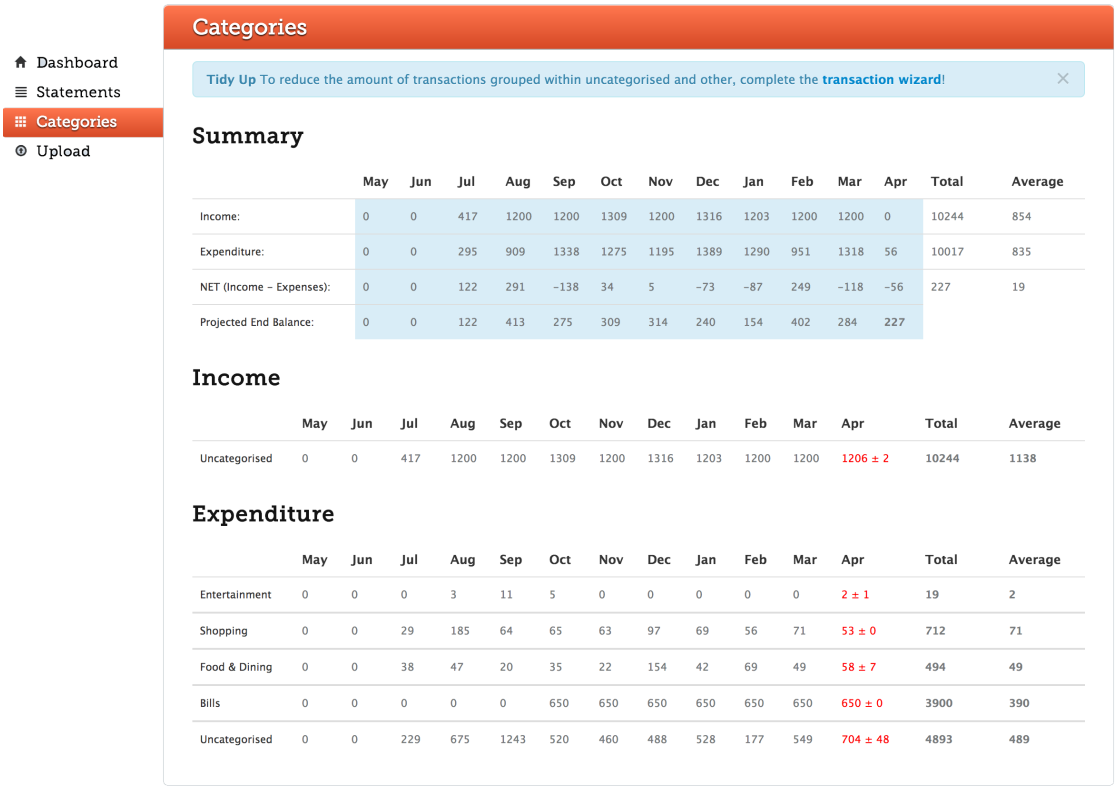
.png

Clicking on any of the rows in the table reveals the subcategories and their associated values that are being used to produce the row, using an animated 'slide-down' effect 5.14.

5.4 Viewing Statements

The application also provides an easy way to view historical spending organised by month (Fig 5.15). The transactions in each month can be grouped by category, transactor or date to give a more detailed idea of how money is being spent (Figs. 5.16 and 5.17).

A user can also view particular transactor or reference, which includes the references category and a summary of recent transactions (Fig. 5.18). If the reference is not yet



Tidy Up To reduce the amount of transactions grouped within uncategorised and other, complete the **transaction wizard**!

Summary

	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Income:	0	0	417	1200	1200	1309	1200	1316	1203	1200	1200	0	10244	854
Expenditure:	0	0	295	909	1338	1275	1195	1389	1290	951	1318	56	10017	835
NET (Income – Expenses):	0	0	122	291	–138	34	5	–73	–87	249	–118	–56	227	19
Projected End Balance:	0	0	122	413	275	309	314	240	154	402	284	227		

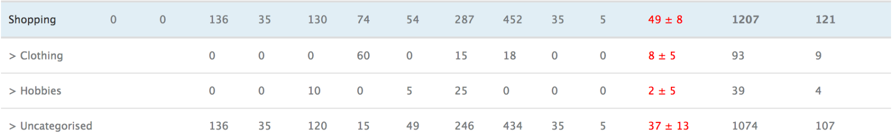
Income

	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Uncategorised	0	0	417	1200	1200	1309	1200	1316	1203	1200	1200	1206 ± 2	10244	1138

Expenditure

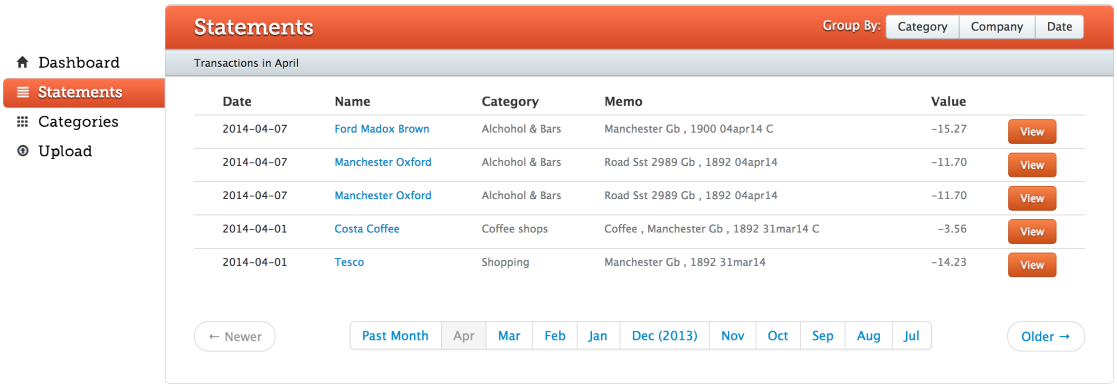
	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Total	Average
Entertainment	0	0	0	3	11	5	0	0	0	0	0	2 ± 1	19	2
Shopping	0	0	29	185	64	65	63	97	69	56	71	53 ± 0	712	71
Food & Dining	0	0	38	47	20	35	22	154	42	69	49	58 ± 7	494	49
Bills	0	0	0	0	0	650	650	650	650	650	650	650 ± 0	3900	390
Uncategorised	0	0	229	675	1243	520	460	488	528	177	549	704 ± 48	4893	489

Figure 5.13: The transaction overview screen, which shows expenditure per month and a prediction (in red) for next month



Shopping	0	0	136	35	130	74	54	287	452	35	5	49 ± 8	1207	121
> Clothing			0	0	0	60	0	15	18	0	0	8 ± 5	93	9
> Hobbies			0	0	10	0	5	25	0	0	0	2 ± 5	39	4
> Uncategorised			136	35	120	15	49	246	434	35	5	37 ± 13	1074	107

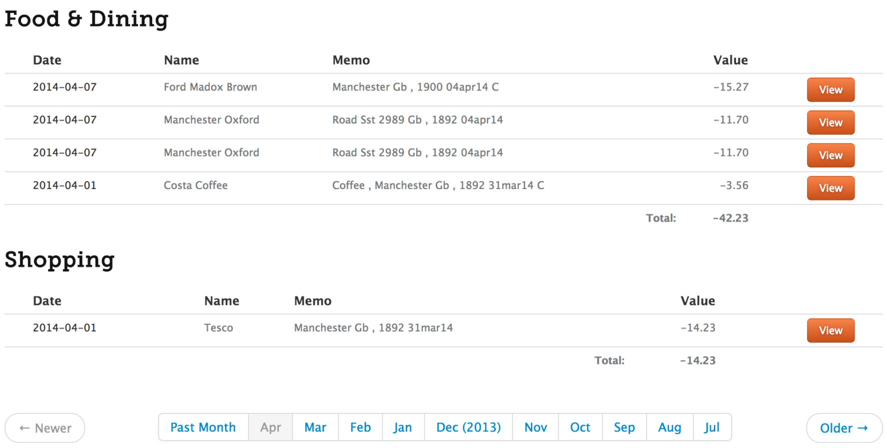
Figure 5.14: The subcategories being used to make up a category in the overview



Transactions in April

Date	Name	Category	Memo	Value	
2014–04–07	Ford Madox Brown	Alchohol & Bars	Manchester Gb , 1900 04apr14 C	–15.27	View
2014–04–07	Manchester Oxford	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	–11.70	View
2014–04–07	Manchester Oxford	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	–11.70	View
2014–04–01	Costa Coffee	Coffee shops	Coffee , Manchester Gb , 1892 31mar14 C	–3.56	View
2014–04–01	Tesco	Shopping	Manchester Gb , 1892 31mar14	–14.23	View

Figure 5.15: The statement view



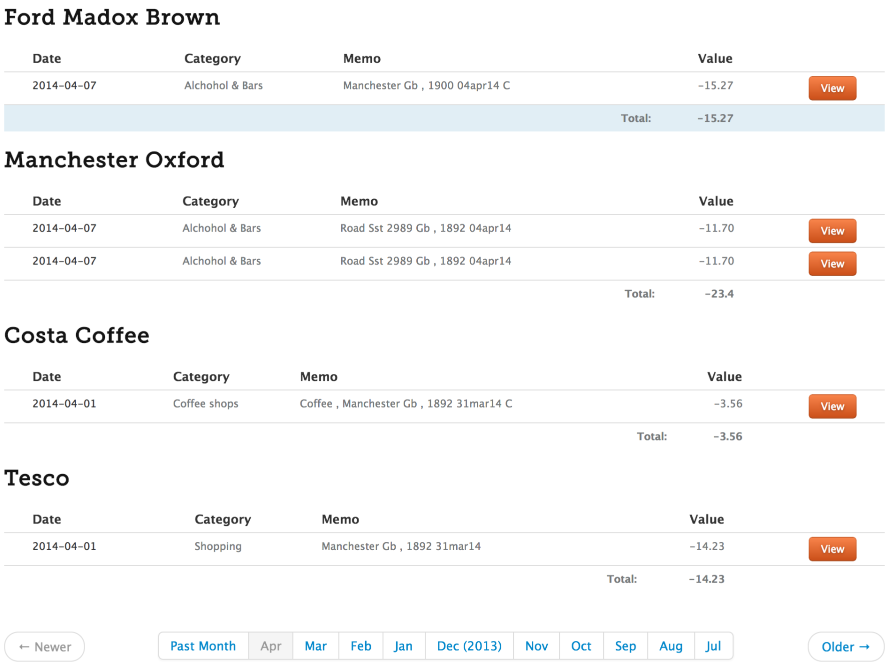
Food & Dining

Date	Name	Memo	Value	
2014-04-07	Ford Madox Brown	Manchester Gb , 1900 04apr14 C	-15.27	View
2014-04-07	Manchester Oxford	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
2014-04-07	Manchester Oxford	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
2014-04-01	Costa Coffee	Coffee , Manchester Gb , 1892 31mar14 C	-3.56	View
		Total:	-42.23	

Shopping

Date	Name	Memo	Value	
2014-04-01	Tesco	Manchester Gb , 1892 31mar14	-14.23	View
		Total:	-14.23	

Figure 5.16: Grouping the statement by category



Ford Madox Brown

Date	Category	Memo	Value	
2014-04-07	Alchohol & Bars	Manchester Gb , 1900 04apr14 C	-15.27	View
		Total:	-15.27	

Manchester Oxford

Date	Category	Memo	Value	
2014-04-07	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
2014-04-07	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
		Total:	-23.4	

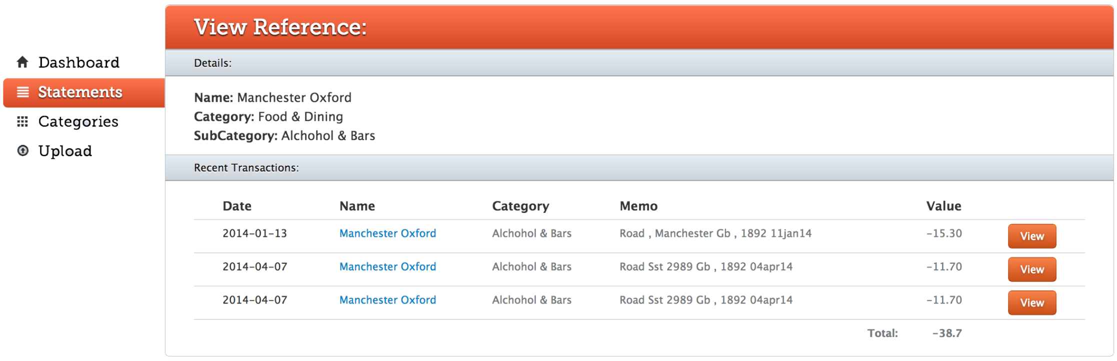
Costa Coffee

Date	Category	Memo	Value	
2014-04-01	Coffee shops	Coffee , Manchester Gb , 1892 31mar14 C	-3.56	View
		Total:	-3.56	

Tesco

Date	Category	Memo	Value	
2014-04-01	Shopping	Manchester Gb , 1892 31mar14	-14.23	View
		Total:	-14.23	

Figure 5.17: Grouping the statement by transactor



View Reference:

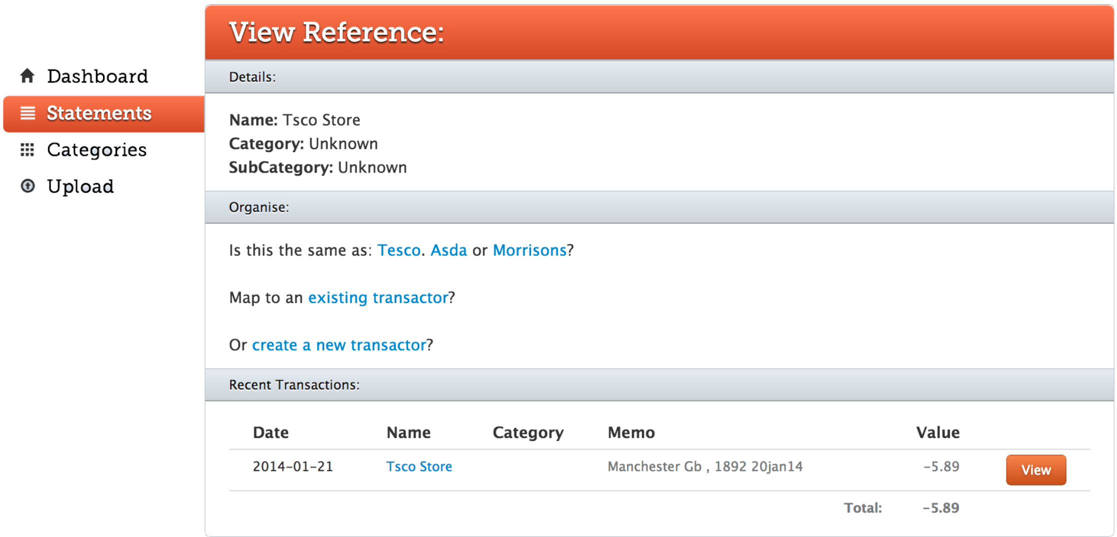
Details:

Name: Manchester Oxford
Category: Food & Dining
SubCategory: Alchohol & Bars

Recent Transactions:

Date	Name	Category	Memo	Value	
2014-01-13	Manchester Oxford	Alchohol & Bars	Road , Manchester Gb , 1892 11jan14	-15.30	View
2014-04-07	Manchester Oxford	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
2014-04-07	Manchester Oxford	Alchohol & Bars	Road Sst 2989 Gb , 1892 04apr14	-11.70	View
			Total:	-38.7	

Figure 5.18: Recent transactions at a particular transactor



View Reference:

Details:

Name: Tsco Store
Category: Unknown
SubCategory: Unknown

Organise:

Is this the same as: Tesco. Asda or Morrisons?

Map to an existing transactor?

Or create a new transactor?

Recent Transactions:

Date	Name	Category	Memo	Value	
2014-01-21	Tsco Store		Manchester Gb , 1892 20jan14	-5.89	View
			Total:	-5.89	

Figure 5.19: Viewing an unmapped reference

mapped, options similar to those shown in the suggestions wizard are listed enabling them to map the reference correctly (Fig. 5.19).

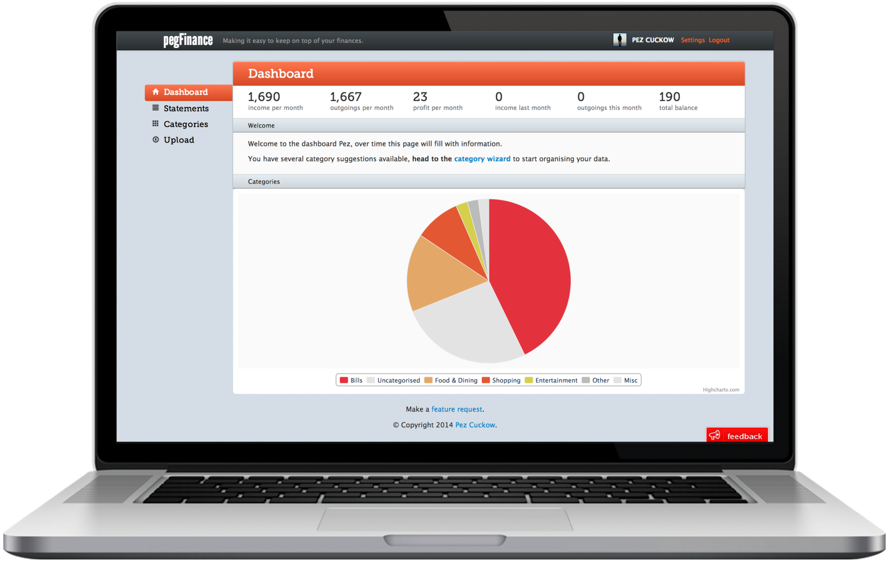
5.5 Responsive Web Design

A key feature of the application is being able to access it at any time from any device, particularly when taking into account the rapid increase in the use of mobile devices. Interacting with a website on a smartphone or tablet is not the same as interacting using a computer, due to the smaller screen size and use of touch over a mouse.

Forbes reported that 24% of their 2013 website visits came from mobiles and 13% from tablets, down from a total of 15% in 2012. particularly with the high percentage of website visits coming from mobile devices [66].

In order to ensure the project is accessible from a variety of different devices the core UI uses Responsive Web Design (RWD) to layout the website differently depending on the screen size of the device to ensure an optimal viewing experience.

The differences depending on the device are highlighted in Figs. 5.20-5.23.



1,690	1,667	23	0	0	190
income per month	outgoings per month	profit per month	income last month	outgoings this month	total balance

Welcome

Welcome to the dashboard Pez, over time this page will fill with information.

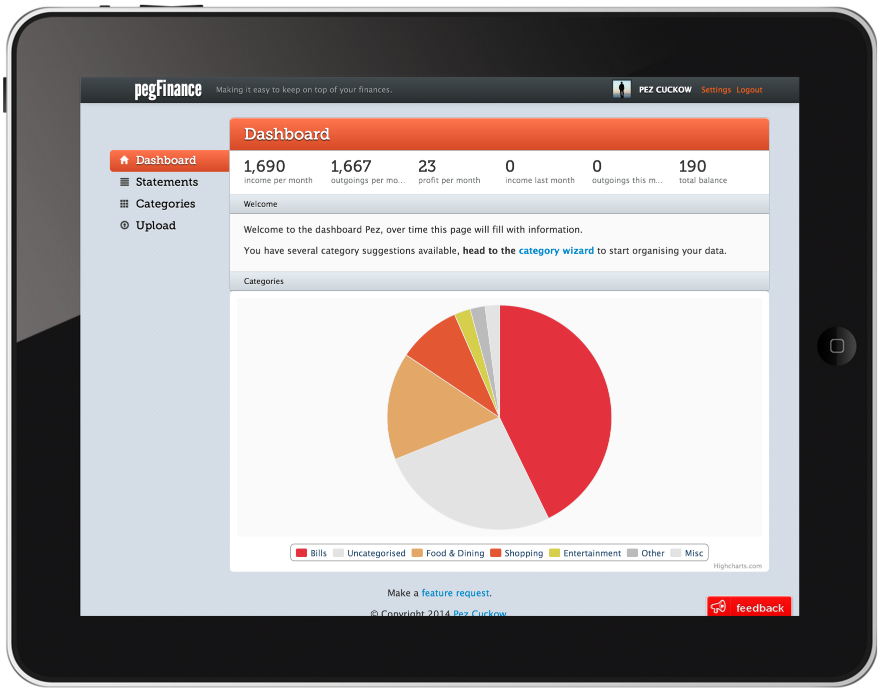
You have several category suggestions available, **head to the** category wizard to start organising your data.

Categories

Make a feature request.

© Copyright 2014 Pez Cuckow.

Figure 5.20: Layout on a standard laptop



Dashboard

1,690 income per month
1,667 outgoings per mo...
23 profit per month
0 income last month
0 outgoings this m...
190 total balance

Welcome

Welcome to the dashboard Pez, over time this page will fill with information.

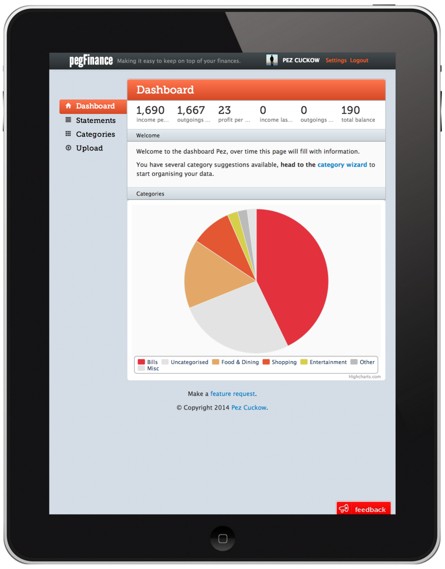
You have several category suggestions available, **head to the category wizard** to start organising your data.

Categories

Make a feature request.

© Copyright 2014 Pez Cuckow

Figure 5.21: Layout on a tablet in landscape



Dashboard

1,690 income pe...
1,667 outgoings ...
23 profit per ...
0 income las...
0 outgoings ...
190 total balance

Welcome

Welcome to the dashboard Pez, over time this page will fill with information.

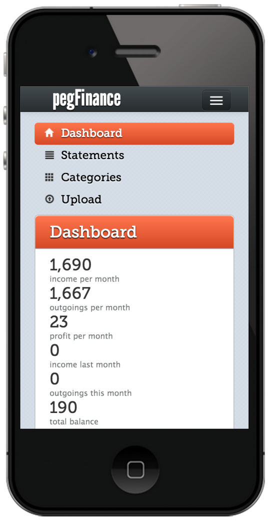
You have several category suggestions available, **head to the category wizard** to start organising your data.

Categories

Make a feature request.

© Copyright 2014 Pez Cuckow.

Figure 5.22: Layout on a tablet in portrait



Dashboard

1,690 income per month
1,667 outgoings per month
23 profit per month
0 income last month
0 outgoings this month
190 total balance

Figure 5.23: Layout on a smartphone

Chapter 6

Testing and Evaluation

The evaluation of the application indicates that these three objectives have been met. Preliminary research demonstrates that enjoyed using the application and that the majority of participants would use it in the future if implemented as a full product. Testing with participant data shows that the average absolute error of the prediction engine was $PERCENTAGE\%$ for users with over three months historical information. Preliminary penetration testing, using both white-box and black-box testing indicated that the security protections put in place are effective. However, the evaluation of the project was limited, due to the size of the user base, discussed further in section 7.1.

6.1 During Development

Though-out the development of the project, predominantly after each feature was implemented and before a new iteration was started the applications functionality was tested through face to face conversations, observations and unit testing.

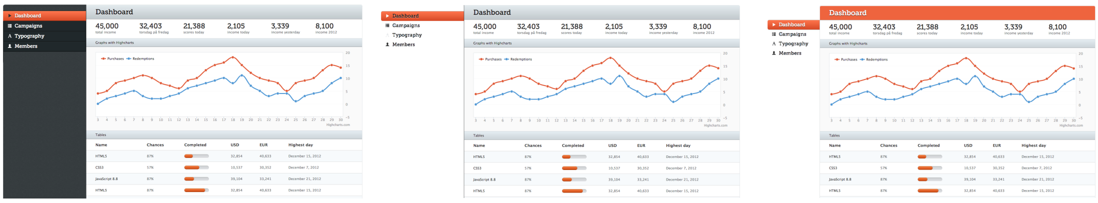
6.1.1 Acceptance Testing

As noted throughout the report, the application was regularly tested by potential end users and the developers during lab observations where the user was asked to complete a task on the website while describing what they were doing.

During the testing, common pain points were identified and these were used to decide on the future features, either to improve the user experience on the website, making it easier to use, or to support a new use case.

Example modifications, made as a result of these observations, include the dropdown menu autocompletion when creating and mapping transactors, the hints that appear throughout the website to guide the user onto the next task, and the suggestion wizard.

Towards the beginning of the project these meetings were used to evaluate and update the design of the user interface, which went through several different designs, and to receive qualitative feedback from the users (Fig. 6.1).



Name	Chances	Completed	USD	EUR	Highest day
HTML5	87%		32,854	40,633	December 15, 2012
CSS3	57%		10,537	30,352	December 7, 2012
JavaScript 8.8	87%		39,104	33,241	December 21, 2012
HTML5	87%		32,854	40,633	December 15, 2012

Figure 6.1: A few iterations of the user interface designs

6.1.2 Unit Testing

High risk classes, including the Budget, the QIF and OFX Parsers implementing the TransactionFileParserInterface, and those implementing the TransactionCollectionInterface were unit tested to ensure the functionality of the classes was not affected when implementing new features or refactoring the code.

These classes were selected as the core functionality of the application relied on their behaviour and they were coupled to the most other classes, for example the Budget object was responsible for holding a collection of Transactons in the tree data structure described in section 4.3 and performed the majority of the prediction functionality, holding references to the TransactionMarkovChain, TransactionWeightedAverageCalculator and PredictionEvaluation.

Due to time constraints and the complications of testing data driven classes, both the number of unit tested classes and methods covered was limited, this is discussed further in section 7.1.

6.2 After Development

Once development of the project was halted the application was tested in the three following the three sections outlined through the report. The statement management functionality was assessed using a questionnaire posed to research participants that were given access to the system, the prediction algorithms were evaluated using the sample data provided by the research participants and the security of the application was evaluated using penetration testing. Unfortunately the reliability of the results for both the statement management and prediction is questionable due to the limited number of research participants and questionnaire responses, see section 7.1.

6.2.1 Statement Management

The questionnaire posed to the research participants was split into three sections, a full copy of the questions can be found in Appendix G.

The first made a series of statements and asked the respondent to answer on a five point scale, ranging from 1, strongly disagree to 5, strongly agree. The questions in this section had four main types, and were selected following the structure of the Standardized Universal Percentile Rank-Questionnaire which is used by companies including

PayPal to assess their websites [67]. Questions on usability focused on how easy users found navigation of the website, locating the information they needed, and whether they enjoyed using it. The potential loyalty of the participants was judged by using the likely hood of a user recommending the site to their colleagues and friends or returning to the site in the future.

The second section focused on task completion, participants were asked to complete a task and then rate the how difficult or easy they found that on a seven point scale. This section was designed to assess the functionality of the website and to identify additional pain points. One scale question was asked per task as research suggests that use of a single question performs just as well as breaking the task down into sub-questions [68], [69], however a free text field was included with each task to obtain qualitative information on the reasons behind the answer.

In final section was used to identify features of the the website that users favoured and those they disliked in order to backup the points reviewed in section two, all questions were answered in free text fields.

In the first section, answered on a 1-5 scale, Loyalty was the question type with the highest average score of 4.10 and a standard deviation of 1.66, which indites some uncertainly, further analysis of the answers, shown in Fig. 6.2, shows that the majority of answers fell between the first and third quartiles and that the median value was 5, indicating that the result is being negatively skewed and affected by outliers at the low end. The statement with the highest average level of agreement was 'If completed, I would visit this site in the future' with an average answer of 4.20, though the standard deviation again quite high at 1.79. Hopefully, this implies that the automation of budget management was of useful to the majority of participants and has potential as a real world product. The usability questions received the lowest score, with an average agreement of 3.50 which is slightly agree. The box plot for the usability questions is shown in Fig. 6.3 which indicates that the mean is a fair measure of this data, though the data is negatively skewed, it also shows that, although the median answer was 4.00 answers ranged between 1 and 5. The questions with the lowest agreement level, overall, were both within usability. 'The website is easy to use' and 'I am able to find what I need quickly' which both scored 3.00, with a standard deviation of 1.41 and 1.22 respectively. On the scale a 3 is neutral, the respondents neither agree or disagree with the statement. The relatively low agreement level of the usability questions was a surprise. It was believed that the hints shown throughout the site and the single navigation menu had improved the usability of the website, however it's clear this area may need extra work.

The responses to the task difficulty questions, shown in Fig. 6.4, indicated that the most difficult tasks to complete were downloading a the users statement from their online banking account and viewing their monthly predictions. Downloading statements may be outside the control of the application, but comments including 'difficult selecting the dates that you want' and 'not sure which download option to choose' indicate that more detailed guidance could be provided to the users when they're told to download their

[1] The whiskers indicate the minimum and maximum values

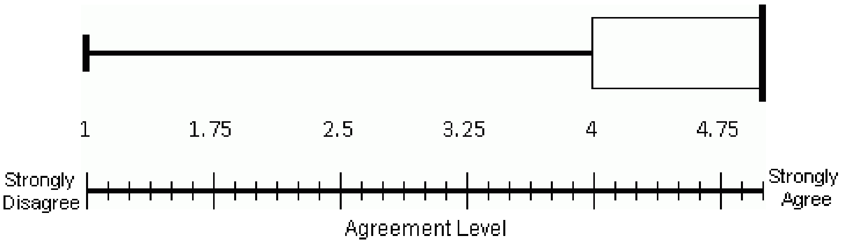


Figure 6.2: Box plot of Loyalty question responses[1]

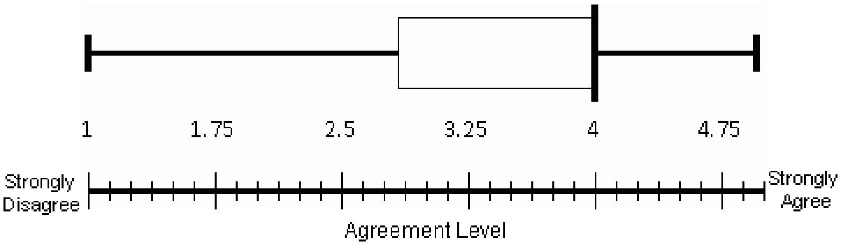


Figure 6.3: Box plot of Loyalty question responses

statement. The relatively low score of the monthly prediction section of the website, is supported with comments including 'too much information' and 'very messy screen', suggesting that further work should be done to improve it's layout and to reduce the amount of detail shown. Overall the responses to the task difficulty questions are positive, with an average difficulty rating of 5.7, indicating that website is relatively easy to use.

The textual responses to the questions in the final section were in line with the task difficulty measures, with difficult tasks noted as the least useful. Interestingly, the pie chart breaking down the expenditure per category was selected as the most useful feature by 100% of the respondents who answered the question, which was matches the background research for existing applications.

6.2.2 Prediction

By taking the test participant data, it was possible to evaluate the prediction accuracy of the application. Users with more than four months history were selected and the data was split into two halves. All but the most recent complete month was fed to the prediction engine and used to make a prediction for total spending and income in the known month. By comparing the recorded predictions to the actual values for that month it was possible to calculate the absolute and percentage error.

Overall the system performed with 16.7% average error, performing with a lower average error of 14.6% for income and a slightly higher error of 18.4%. It is likely that income has a lower error as the income is often regular and for similar amounts. However, as only 7 of the participants had uploaded more than four months data the scope of the testing was limited and the reliability of the results is questionable. In addition, three

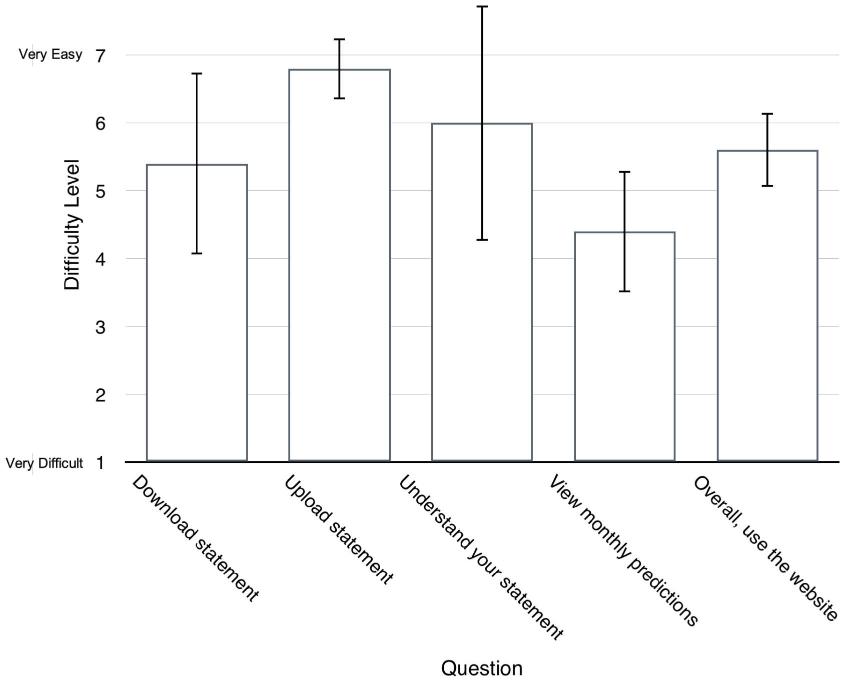


Figure 6.4: Responses to the task difficulty questions, the standard deviation is shown as error bars

of the records were identified as outliers as the absolute error fell significantly outside of the inter quartile range, and not included in the averages. Outliers were identified using $\left[Q_1 - 1.5(Q_3 - Q_1), Q_3 + 1.5(Q_3 - Q_1)\right]$ where Q_1 and Q_3 were the upper and lower quartiles. The full results of the testing data are shown in Table 6.1 and the quartiles are shown in Table 6.2, records that fell outside of the bounds and were ignored when calculating the averages are indicated in italics. It is likely that these outliers were caused by sudden changes in spending pattern or that the prediction is being affected by significant expenditure or income in January.

6.2.3 Security

The security of the application was measured using white-box and black-box penetration testing. The author, who had a working knowledge of the internals of the application and access to the source code, performed the white-box testing. They attempted to break into the account of someone using the website through the use of session hijacking outlined in subsection 2.3.1, even after disabling the encryption of the users session by temporarily allowing allow cookies to be sent over HTTP, and gaining a copy of the cookie the account couldn't be accessed as the users session was invalidated once the session hijack was detected by the fingerprinting mechanism. Even after the headers of the target user were cloned, the users session ID didn't match the one found in the database and the session was once again invalidated. Black-box penetration test-

		February			
User	Type	Prediction (£)	Actual (£)	Absolute Error (£)	Error (%)
1	Income	1620	1409	211	15.0
	Expenditure	1710	2603	893	34.3
2	Income	2080	209	1871	*895.2*
	Expenditure	1678	1633	45	2.8
3	Income	1187	1200	13	1.1
	Expenditure	1227	943	284	30.1
4	Income	864	881	17	1.9
	Expenditure	659	814	155	19.0
5	Income	1060	732	328	44.8
	Expenditure	1101	1309	208	15.9
7	Income	243	100	143	*143.0*
	Expenditure	383	51	332	*651.0*
8	Income	1008	914	94	10.3
	Expenditure	1113	1029	84	8.2
				Average	**13.7**

Table 6.1: The results of the prediction evaluation test, which compared predictions and actual values

	Quartile	Bound
Lower Bound	8.7	-172.6
Second Quartile	17.5	
Upper Bound	111.5	265.8

Table 6.2: First, second and third quartiles for the prediction evaluation test

ing was performed by a peer of the author, who has experience in penetration testing. The unsucesfully tried SQL injection by modifying the parameters sent in the suggestion wizards AJAX requests before uploading a bank statement that contained cross site-scripting (XSS) code designed to steal the users cookie. The uploaded statement was parsed successfully and all of the XSS was escaped, preventing the code injection. Finally, the tester attempted to upload malicious files in an attempt to break the upload feature, in all cases the Although this demonstration isn't able to conclude that the application is secure and resilient to all attacks, it does demonstrate that more common attacks including those outlined in section 2.3 are prevented.

Chapter 7

Conclusions

The project set out to build an application that made personal finances easier to manage through the automation of the common steps that people go through when making a budget.

Existing finance applications were researched, highlighting the advantages and disadvantages of each, as well as identifying the features that users found useful through the use of reviews. A key piece of the functionality, the prediction engine, was discussed, investigating a combination of existing techniques used to forecast financial spending, including MCMs and Weighted Arithmetic Means. The ethical implications of storing high risk personally identifiable information was assessed and considered with reference to application security and strong security practice. This research was then used to decide on the functionality of the application, the design and implementation of the prediction engine and used to ensure high security standards were met throughout the application.

Specific design and implementation challenges were discussed, highlighting the key background and technical knowledge that was gained in order to overcome the challenges faced.

The aim of the project was to build an application that implemented an intuitive way to view and manage personal finances and accurately predict a users future transactions based on their history, while upholding the high security expectations of such a service. The final application was reviewed, following the usage pattern of the average user, to outline the key features implemented and to give an idea of how it works.

The evaluation of the application indicates that these three objectives have been met. Preliminary research demonstrates that users enjoyed using the application and indicates the majority of participants would use it in the future if released as a full product. Testing with participant data shows that the average percentage error of the prediction engine was 16.7% for users with over four months historical information. Preliminary penetration testing, using both white-box and black-box testing indicated that the security protections put in place are effective. However, the evaluation of the project was limited, due to the size of the user base, discussed further in section 7.1.

7.1 Limitations

The main limitation of the project was the scope of the test participants. A total of 20 testers signed up to participate in the project during it's development, uploading at least one month of transaction data. However, only 7 of these individuals uploaded more than four months of transaction history, which was the threshold for using training and testing to select a best fit weighting model. In addition the majority of the participants were peers of the author, which meant that the majority of evaluation and testing was performed using students.

More extensive testing using a broader spectrum of testers, who may require a variety of different features, or use the system in a different way, may provide a greater level of evaluation for the system.

A key difficulty when seeking participants were their concerns regarding data security and trusting the researchers with their personal information. It it hoped that by publishing the level of concern and detail that has gone into the design of the security of the application, including encryption of their personally identifiable information, that the effects of this concern can be limited.

If the project were to continue, a more open approach for recruiting participants could be taken, such as seeking alpha testers and feedback in online forums and news groups. This should lead to both larger numbers of participants and a wider variety in the users background to give a fairer picture of potential users. Due to the variety of formatting already seen in participant data, it may be appropriate to limit the participants to the UK. A continued concern would be the security of the application when opening it up to users on the Internet, therefore this would need to be addressed.

During the testing and evaluation there were many features suggested by users to improve the application to their individual preferences. Due to time restraints the majority of these were not implemented and it was decided it was more imperative to focus on the most important and most frequently requested features first. Further development could include the additional requested features to suit a greater variety of users.

Although the key functionality of the application was unit tested, tests of the business logic in the prediction engine and on key objects was limited. This limitation was due to the heavy reliance on data, and because the engine performs differently depending on the users history. To test this functionality in the future data with pre-calculated attributes would be need to be prepared and a separate database access for storing this testing data would need to be implemented.

7.2 Further research

Due to the nature and size of this project a wide variety of further research opportunities have been highlighted.

7.2.1 Model Selection

Currently, the application selects most appropriate weighting model to use for each users transactions by evaluating the pre-defined weighting models via their historical data, and selecting the one with the least total overall error.

User Classification

Given the wealth of information the application holds about the users, higher accuracy could potentially be achieved by classifying each user. The users could be classified in two different ways; using pre-defined groups, or using cluster analysis.

To split the users into existing groups, the researchers could identify groups that they believed would exhibit similar spending patterns and characteristics. Users would identify which group they felt they fit into when signing up and this information would be used to define the characteritics of each group. Having chosen features that identify each group, the researchers could automatically classify the users without prompting them

Alternatively, the use of cluster analysis could be employed to determine the number of groups, as well as the members of groups. The objective of the clustering would be to group users that exhibited similar spending patterns together, so those that were most similar were together and more similar to those in other clusters.

Having classified the users, the parameters in the prediction engine could be configured for each individual group rather than by user. In addition, it would allow similar users to be compared, which would allow statements such as 'users similar to you spend, on average, 10% more' to be made, and could perhaps be used to make savings suggestions to each user.

Further research could investigate the reliability of the classification using different clustering algorithms and whether configuring the prediction engine per group leads to better results than per user.

Model Selection

Weighting models were currently selected by category. The users total transactions per month in a particular category were evaluated and the weighting model with the best fit is chosen for that category. Transactions were considered by category and by month. Data was split by category which was chosen as the number of transactions in each subcategory was low, and preliminary experimentation showed that sub-categories would suit the same model as their category. Using months as the time period was selected as the majority of participant data contained regular monthly transactions such as household and mobile phone bills, and so considering the transactions at smaller time steps led to errors in the Markov Chain Model as the probability of a transaction not occurring increased.

However, different time periods and methods of splitting the data could be investigated. For example transactions at each individual transactor could be assigned their own model, or all spending below a certain value could use the same model. The time

$$\bar{x} = \frac{\sum_{t=0}^{n-1} w(t) \times x_t}{\sum_{t=0}^{n-1} w(t)}$$

Figure 7.1: Weighted arithmetic mean

periods used to build the model, could be by month, weeks, days or perhaps even by day of the week. Use of a time period such as a day of the week would result in a more complected Markov Chain Model where samples such as 'given that the last purchase was on a Wednesday' could be taken.

Further research could investigate the effects of using different grouping methods and time periods on the overall accuracy of the predictions.

7.2.2 Alternative Forecasting Techniques

The system currently uses techniques adapted from exponential smoothing to forecast and predict the value of a future transaction. Currently this is limited to a weighted arithmetic mean in the form shown in Fig. 7.1. Use of a mean to forecast the value of the next period performs poorly when there is a trend in the data [21].

Second order or double exponential smoothing can be used to take into account trends and seasonal changes in data, such as the average value of an individuals weekly shop increasing over time, or the increase of purchases during Christmas. Double exponential smoothing involves two steps, calculating the weighted mean as before, and then adjusting that mean by the difference between the previous value and it's associated prediction, shown in Fig. 7.2, the comparison between the weighted mean and second order forecasting transaction that is increasing in value by 10, for each time period is shown in Fig. 7.3. Triple exponential smoothing extends the trend calculation to include a seasonal index. Assuming at least one complete seasons data is available the value of x_t of scaled by this seasonal index, which is calculated using by taking the average value of this period historically and dividing it by the average value for the year that the period occurred or $S_t = \frac{x_t}{\bar{x}_p}$ and $\bar{x}_p = \frac{\sum_{i=1}^{p} x_i}{p}$ where p is the number of periods in a year, and x_i is the value of the period i in the year considered. Fig. 7.4 shows the performance on some data that follows a similar trend to the previous year [20], [22].

Further research could investigate the effect of applying double order and triple exponential smoothing to forecast the values of transactions on the final prediction and how seasonal indexes could be calculated. Seasonal indexes, for example, could be calculated per transactor, category, user or globally across the application. The research would need access to historical transaction data for the users to be able to make these assessments.

7.2.3 Learning the Scaling Parameters

It was mentioned in section 3.2.3 that the speed of the decay in the weighting functions could adjusted by using a scaling constant. Instead of choosing the best weighting model from a pre-defined selection, the application could apply non-linear optimisation

$$\bar{x}_t = \alpha(t) \times \bar{x}_t + (1 - \alpha(t))(\bar{x}_{t-1} + b_t - 1)$$

$$b_t = \alpha(t)(\bar{x}_t - \bar{x}_{t-1}) + (1 - \alpha(t))b_{t-1}$$

Where $\alpha(t) = \frac{w(t)}{\sum_{t=0}^{n-1} w(t)}$ representing the final weight
In general, the starting values are set as $\bar{x}_1 = x_1$ and $b_1 = x_2 - x_1$

Figure 7.2: Calculating the second order weighted arithmetic mean

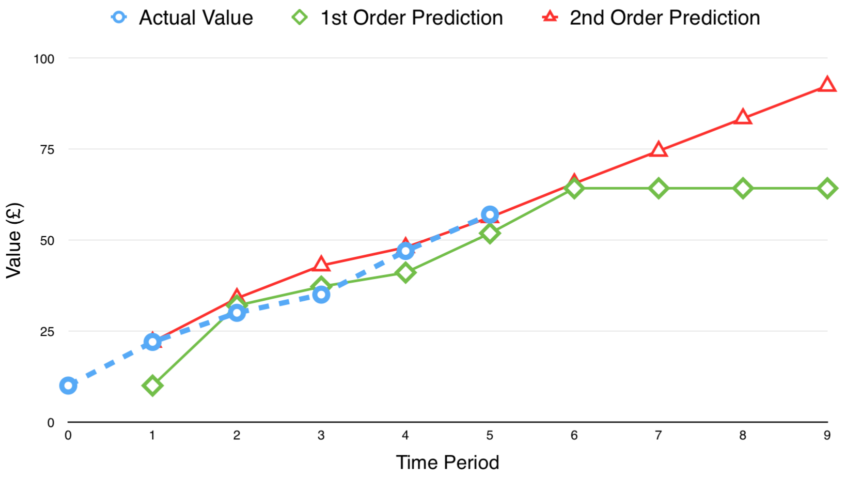


Figure 7.3: Comparison of first and second order forecasting when predicting a trend

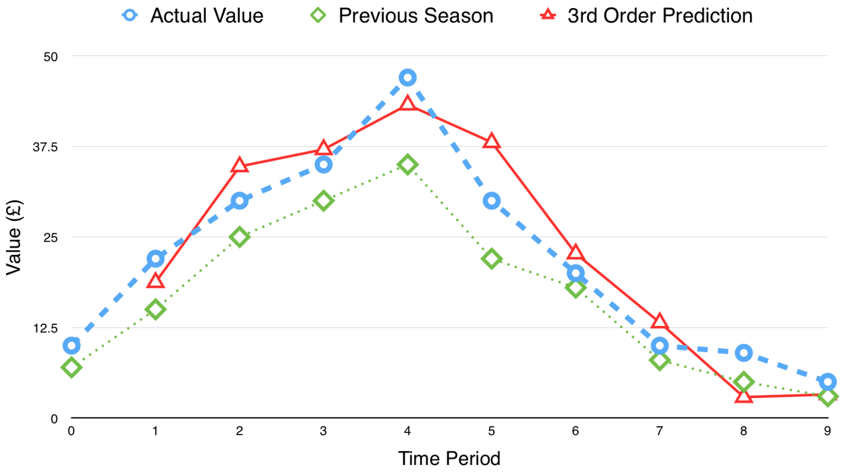


Figure 7.4: Using third order forecasting on data with a previous season

techniques to adjust the rate of the decay, and select a customised model for each user.

Further research could investigate different implementations of these optimisation techniques, and how the use of custom models affects the prediction accuracy for each user.

References

[1] C. Gore, “The global recession of 2009 in a long-term development perspective,” *Journal of International Development*, vol. 22, no. 6, pp. 714–738, 2010.

[2] A. Barnard, “The economic position of households, q1 2012,” *Office Of National Statistics*, 2012. [Online]. Available: `http://www.ons.gov.uk/ons/rel/hsa/the-economic-position-of-households/q1-2012/art---the-economic-position-of-households--q1-2012.html` (visited on 04/08/2014).

[3] The Wall Street Journal. (2013). The experts: is creating a personal budget a good idea?

[4] Think Local Act Personal. (2013). 2nd national personal budget survey. L. Boyd, Ed., [Online]. Available: `http://www.thinklocalactpersonal.org.uk/Latest/Resource/?cid=9503`.

[5] BBC News. (2010). Debit card spending in uk overtakes cash for first time, [Online]. Available: `http://www.bbc.co.uk/news/business-11901953` (visited on 04/09/2014).

[6] Lloyds Bank. (2014). Lloyds bank - private banking - internet banking, [Online]. Available: `http://www.lloydsbank.com/private-banking/contact-us/internet-banking.asp` (visited on 04/20/2014).

[7] Mint. (2014). What is mint, Intuit, Inc, [Online]. Available: `https://www.mint.com/what-is-mint/` (visited on 04/20/2014).

[8] Lloyds Bank, Ed. (2014). Money manager, The free, easy way to keep track of your money, [Online]. Available: `http://www.lloydsbank.com/online-banking/benefits-online-banking/money-manager.asp` (visited on 04/09/2014).

[9] Money Watch. (2011). Lloyds tsb money manager, [Online]. Available: `http://money-watch.co.uk/7992/lloyds-tsb-money-manager` (visited on 04/18/2014).

[10] ‘Deals’ *et al.* (2011). Anyone tried money manager at lloyds tsb? MoneySavingExport.com Forum, [Online]. Available: `http://forums.moneysavingexpert.com/showthread.php?t=3114854` (visited on 04/21/2014).

[11] Stack Overflow Users. (2010). Banking api/protocol, Stack Overflow, [Online]. Available: `http://stackoverflow.com/questions/3469628/banking-api-protocol` (visited on 04/19/2014).

[12] ——, (2010). Is there an api to get bank transaction and bank balance? Stack Overflow, [Online]. Available: `http://stackoverflow.com/questions/7269668/is-there-an-api-to-get-bank-transaction-and-bank-balance` (visited on 04/20/2014).

[13] K. Purcell, "Half of adult cell phone owners have apps on their phones," *Pew Internet & American Life Project*, 2011.

[14] iTunes. (2013). Top 10 apps - finance, Apple, [Online]. Available: `http://www.apple.com/euro/itunes/charts/apps/top10appstorefinance.html` (visited on 04/20/2014).

[15] SpendeeApp, Ed. (2014). Spendee - see where your money goes, [Online]. Available: `http://www.spendeeapp.com` (visited on 04/09/2014).

[16] S. Flückiger. (2013). Budgt, [Online]. Available: `http://www.spendeeapp.com` (visited on 04/09/2014).

[17] BlueTags, Ed. (2014). Pocket expense, [Online]. Available: `http://www.bluetgs.com/Home.aspx` (visited on 04/09/2014).

[18] P. E. Pfeifer and R. L. Carraway, "Modeling customer relationships as markov chains," *Journal of Interactive Marketing*,

[19] V. Singh, L. Freeman, B. Lepri, and A. Pentland, "Predicting spending behavior using socio-mobile features," in *Social Computing (SocialCom), 2013 International Conference on*, 2013, pp. 174–179. DOI: `10.1109/SocialCom.2013.33`.

[20] S. Dash, "A comparative study of moving averages: simple, weighted and exponential," Applied Technical Analysis, Tech. Rep., 2012. [Online]. Available: `https://www.tradestation.com/education/labs/analysis-concepts/a-comparative-study-of-moving-averages` (visited on 04/17/2014).

[21] J. Filliben *et al.*, "Introduction to time series analysis," in *NIST/SEMTECH Handbook of Statistical Methods*, National Institute of Standards and Technology, 2003.

[22] P. S. Kalekar, "Time series forecasting using holt-winters exponential smoothing," *Kanwal Rekhi School of Information Technology*, vol. 4329008, pp. 1–13, 2004.

[23] J. Williams, A. Jex, *et al.* (2011). Session hijacking attack, Open Web Application Security Project (OWASP), [Online]. Available: `https://www.owasp.org/index.php/Session_hijacking_attack` (visited on 04/22/2014).

[24] E. Butler, "Hey web 2.0: start protecting user privacy instead of pretending to," Presented at Toorcon 12 San Diego, 2010. [Online]. Available: `http://codebutler.github.io/firesheep/tc12/` (visited on 04/19/2014).

[25] ——, (2014). Codebutler/firesheep, GitHub, [Online]. Available: `https://github.com/codebutler/firesheep` (visited on 04/22/2014).

[26] D. Walker-Morgan. (2012). Sniffer tool displays other people's whatsapp messages: news and features, The H Security, [Online]. Available: `http://www.h-online.com/news/item/Sniffer-tool-displays-other-people-s-WhatsApp-messages-1574382.html` (visited on 04/21/2014).

[27] K. Helkala and E. Snekkenes, “A method for ranking authentication products,” in *Proceedings of the Second International Symposium on Human Aspects of Information Security & Assurance*, 2008, ISBN: 9781841021898.

[28] W. Burr, D. Dodson, E. Newton, R. Perlner, W. Polk, S. Gupta, and E. Nabbus, “Electronic authentication guideline,” *National Institute of Standards and Technology*, 2013, Special Publication 800-63-2.

[29] M. Weir, S. Aggarwal, M. Collins, and H. Stern, “Testing metrics for password creation policies by attacking large sets of revealed passwords,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS ’10, Chicago, Illinois, USA: ACM, 2010, pp. 162–175, ISBN: 978-1-4503-0245-6. DOI: `10.1145/1866307.1866327`. [Online]. Available: `http://doi.acm.org/10.1145/1866307.1866327`.

[30] (2012). Linkedin passwords leaked by hackers, BBC News, [Online]. Available: `http://www.bbc.co.uk/news/technology-18338956` (visited on 04/19/2014).

[31] D. Chechik. (2013). Look what i found: moar pony! TrustWave Security Firm, [Online]. Available: `http://web.archive.org/web/20131208203540/http://blog.spiderlabs.com/2013/12/look-what-i-found-moar-pony.html` (visited on 04/22/2014).

[32] M. W. Jorgensen. (2012). Free rainbow tables - distributed rainbow table generation, Distributed Rainbow Table Project, [Online]. Available: `https://www.freerainbowtables.com/tables/` (visited on 04/22/2014).

[33] R. Morris and K. Thompson, “Password security: a case history,” *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979.

[34] Quicken Support. (2010). Quicken interchange format (qif) files, [Online]. Available: `http://web.archive.org/web/20100203121050/http://web.intuit.com/support/quicken/docs/d_qif.html` (visited on 04/10/2014).

[35] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” in *Soviet physics doklady*, vol. 10, 1966, p. 707.

[36] M. May, “Inaccessibility of captcha,” *Alternatives to Visual Turing Tests on the Web, W3C, Editor*, W3C, 2005.

[37] S. Hegarty. (2012). The evolution of those annoying online security tests, BBC News, [Online]. Available: `http://www.bbc.co.uk/news/magazine-18367017` (visited on 04/20/2014).

[38] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” *CoRR*, vol. abs/1312.6082, 2013.

[39] 9kw.eu. (2014). Captcha service for the user - captcha solver, [Online]. Available: `http://www.9kw.eu/index.html` (visited on 04/22/2014).

[40] D. Danchev. (2014). Google's recaptcha under automatic fire from a newly launched recaptcha-solving/breaking service, Webroot Threat Blog, [Online]. Available: `http://www.webroot.com/blog/2014/01/21/googles-recaptcha-automatic-fire-newly-launched-recaptcha-solving-breaking-service/` (visited on 04/21/2014).

[41] I. Savinkin. (2013). 8 best captcha solvers, [Online]. Available: `http://scraping.pro/8-best-captcha-solving-services-and-tools/` (visited on 04/20/2014).

[42] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Fast Software Encryption*, Springer, 1994, pp. 191–204.

[43] National Institute Of Standards And Technology, "Announcing the advanced encryption standard (aes)," 2001.

[44] R. M. Stair and G. W. Reynolds, *Principles of Information Systems: a Managerial Approach*, 9th ed. South-Western, 2009, p. 245, ISBN: 0324665288.

[45] C. Larman, *Applying uml and patterns: an approach to object-oriented analysis and design*, 1997.

[46] Cake Software Foundation. (2014). Understanding model-view-controller, [Online]. Available: `http://book.cakephp.org/2.0/en/cakephp-overview/understanding-model-view-controller.html` (visited on 04/27/2014).

[47] GitHub, Inc. (2014). Pez cuckow - github, [Online]. Available: `https://github.com/pezmc/` (visited on 04/25/2014).

[48] F. Potencier. (2009). Templating engines in php, [Online]. Available: `http://fabien.potencier.org/article/34/templating-engines-in-php` (visited on 04/26/2014).

[49] W. Durand, M. J. Schmidt, R. Dupret, P. Borreli, *et al.* (2014). Propel, the blazing fast open-source php 5.4 orm, [Online]. Available: `http://propelorm.org/` (visited on 04/27/2014).

[50] M. Otto and J. Thornton. (2014). Bootstrap, Twitter, [Online]. Available: `http://getbootstrap.com/` (visited on 04/27/2014).

[51] D. KIZLER. (2013). Pros and cons of bootstrap, [Online]. Available: `http://www.hyperarts.com/blog/twitter-bootstrap-sucks-and-its-awesome/` (visited on 04/26/2014).

[52] I. Carrico. (2013). You don't need bootstrap, [Online]. Available: `http://fourkitchens.com/blog/2013/10/09/you-dont-need-bootstrap` (visited on 04/26/2014).

[53] The jQuery Foundation. (2014). Jquery - write less do more, [Online]. Available: `http://jquery.com/` (visited on 04/27/2014).

[54] Stack Overflow Users. (2013). How to parse a ofx file in php? Stack Overflow, [Online]. Available: `http://stackoverflow.com/questions/15735330/how-to-parse-a-ofx-version-1-0-2-file-in-php` (visited on 04/28/2014).

[55] S. Hanselman. (2006). Postprocessing autoclosed sgml tags with the sgmlreader, [Online]. Available: `http://www.hanselman.com/blog/Postprocessing-Auto-Closed-SGML-Tags-With-The-SGML-Reader.aspx` (visited on 04/28/2014).

[56] GNU Project. (2007). The gnu general public license v3.0, Free Software Foundation, [Online]. Available: `http://www.gnu.org/licenses/gpl.html` (visited on 04/25/2014).

[57] Z. Huber, H. Perrin, *et al.* (2014). Pnotify, SciActive, [Online]. Available: `http://sciactive.github.io/pnotify/` (visited on 04/24/2014).

[58] MySQL. (2014). Natural language full-text searches, Oracle, [Online]. Available: `https://dev.mysql.com/doc/refman/5.0/en/fulltext-natural-language.html` (visited on 04/25/2014).

[59] T. Unger, M. Mulazzani, D. Fruhwirt, M. Huber, S. Schrittwieser, and E. Weippl, "Shpf: enhancing http(s) session security with browser fingerprinting," in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, 2013, pp. 255–261. DOI: `10.1109/ARES.2013.33`.

[60] Google. (2014). Update google chrome, [Online]. Available: `https://support.google.com/chrome/answer/95414` (visited on 04/26/2014).

[61] A. Wyman *et al.* (2014). Update firefox to the latest version, Mozilla, [Online]. Available: `https://support.mozilla.org/en-US/kb/update-firefox-latest-version` (visited on 04/27/2014).

[62] The PHP Group, Ed., *Session Functions, session_unset*. [Online]. Available: `http://www.php.net/manual/en/function.session-unset.php` (visited on 04/27/2014).

[63] The PHP Group, Ed., *Session Functions, session_destroy*. [Online]. Available: `http://www.php.net/manual/en/function.session-destroy.php` (visited on 04/27/2014).

[64] J. Atwood. (2013). The definitive guide to form based website authentication, Stack Overflow, [Online]. Available: `http://stackoverflow.com/questions/549/the-definitive-guide-to-form-based-website-authentication` (visited on 04/27/2014).

[65] T. Preston-Werner *et al.* (2014). Gravatar - globally recognized avatars, [Online]. Available: `https://en.gravatar.com/` (visited on 04/27/2014).

[66] J. Steimle. (2013). Why your business needs a responsive website before 2014, [Online]. Available: `http://www.forbes.com/sites/joshsteimle/2013/11/08/why-your-business-needs-a-responsive-website-before-2014/` (visited on 04/06/2014).

[67] J. Sauro, "The standardized universal percentile rank questionnaire (supr-q)," 2011. [Online]. Available: `http://www.suprq.com/` (visited on 04/28/2014).

[68] J. Sauro and J. S. Dumas, "Comparison of three one-question, post-task usability questionnaires," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2009, pp. 1599–1608.

[69] J. Sauro and J. R. Lewis, "Correlations among prototypical usability metrics: evidence for the construct of usability," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2009, pp. 1609–1618.

Appendices

Appendix A

Survey

Informal survey of 12 CS students in the third year lab.
Questions:

1. Do you currently make a budget?
2. Do you stick to that budget?
3. Do you find your budget has a ‘positive’ impact?

Table A.1: Survey Results

	Question		
Answer	1.	2.	3.
yes	5	1	3
no	7	4	4
n/a	0	7	5

Appendix B

Hashing Test

Implemented in PHP, the test was run on a computer with a 2.7 Ghz Intel Core i7, 16Gb of 1600Mhz DDR3 RAM and PHP CLI 5.5.3.

```
<?php

$timeTarget = 1;
ini_set('memory_limit', '-1');

$strings = array();
for($i = 0; $i < 3000000; $i++) {
  $strings[$i] = randomString(16);
}

$hashingFunctions = array('MD5', 'SHA1', 'BCRYPT');

foreach($hashingFunctions as $hash) {
  for($i = 0; $i < 10; $i++) {
    $start = microtime(true);
    $count = 0;
    do {
      $count++;

      if($hash == $hashingFunctions[0])
        md5($strings[$count]);
      elseif($hash == $hashingFunctions[1])
        sha1($strings[$count]);
      elseif($hash == $hashingFunctions[2])
        password_hash($strings[$count], PASSWORD_BCRYPT);
      else
        echo "ERROR:␣Unknown␣function";

      $end = microtime(true);
    } while (($end - $start) < $timeTarget);
    echo "$hash\t$count\n";
  }
}

function randomString($length = 10) {
    $characters = '0123456789
        abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, strlen($characters) - 1)];
    }
    return $randomString;
}
```

Appendix C

Database Schema

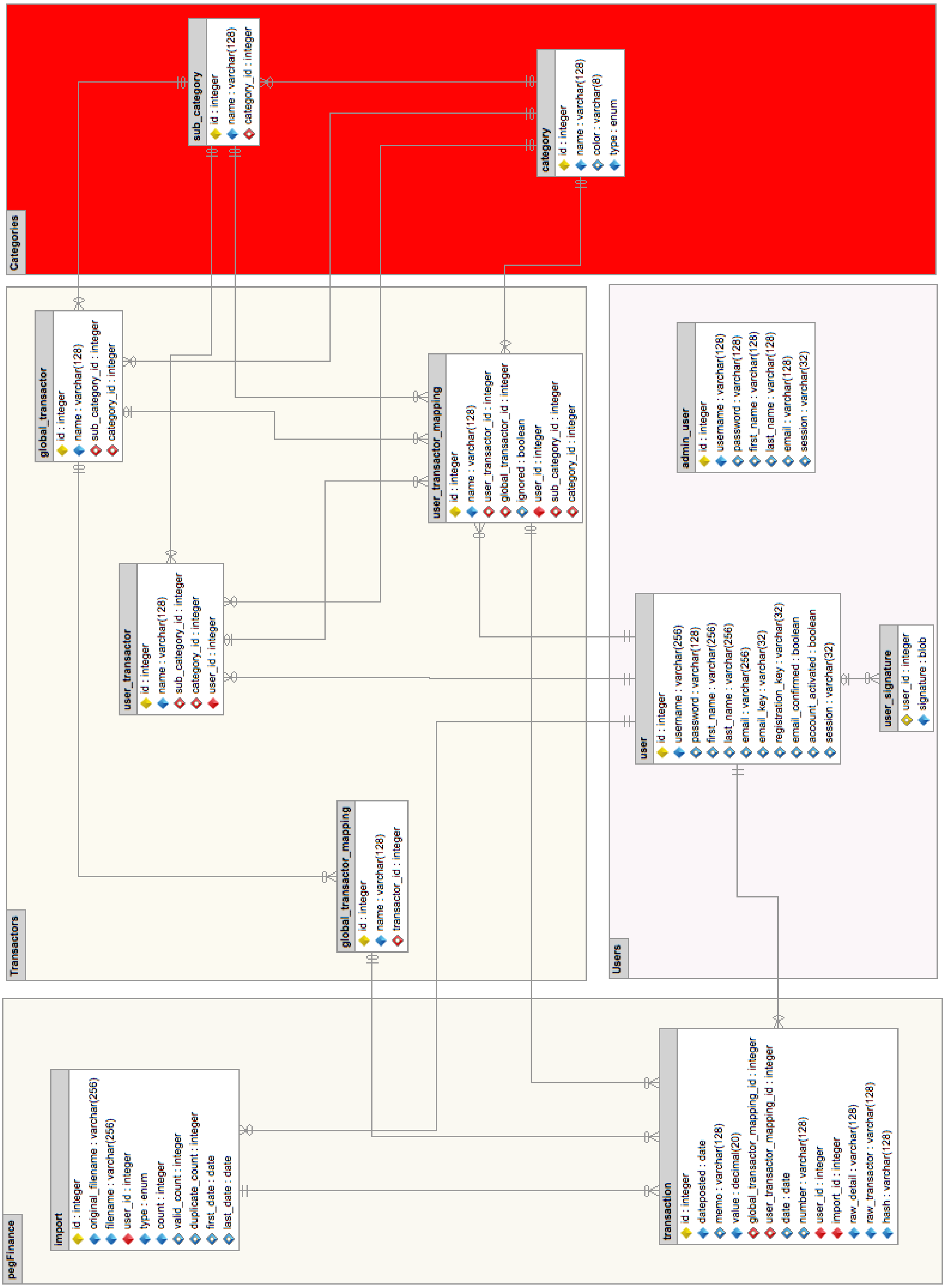


Figure C.1: Full database schema for the project

Appendix D

External Libraries

A full list of libraries and frameworks used by pegFinance are included below, includng licensing information.

D.1 Back End

pegFramework

MVC framework that pegFinance is based around.

Licence - self developed, open-source under the MIT license

Propel

ORM library for PHP, that converts schemas based in XML into PHP objects and provides database connectivity. The database objects found within pegFinance are defined in, and generated by Propel.

License - open-source project released under the MIT license

Less

CSS pre-processor that extends the CSS, adding additional features and functionality such as inheritance. '.Less' files are compiled to '.css' using Less. The CSS for pegFinance is written in and compiled with Less.

License - open-source under the Apache 2 License

D.2 Front End

jQuery

A 'feature-rich JavaScript library', that provides element selection and enabled HTML traversal and manipulation, as well as providing an easy API for Ajax requests. Used throughout the site for frontend effects, and used heavily in the suggestion wizard.

License - open-source, released under the MIT license

Bootstrap

Front-end framework for faster web development that provides a large collection of predefined CSS classes and elements for use on the page. Used for the UI of pegFinance.

License - MIT license and copyright 2014 Twitter

Inspiritas

A theme for bootstrap, that defines colors and additional elements for use on the page. A heavily modified version is used for the UI of pegFinance.

Licence - Open-source, Apache License

StrongPass.js

jQuery Password Strength plugin for Twitter Bootstrap. Modified version used to provide a client side indication the password entropy calculation during signup.

License - Dual, under the MIT and GPL.

Pines Notify

jQuery Pines Notify Plugin. Used to provide event notification in the user interface to indicate events. pNotify is used in the suggestion wizard and when mapping a reference to a transactor to update the user on the status of the request

Licence - Triple licensed under the GPL, LGPL, and MPL.

UserVoice JavaScript SDK

SDK for access to a user opinion platform, to help provide insight into the userbase and receive feedback. Deployed throughout the application so users can provide feedback when using the application, including an optional screen capture.

Licence - Commercial SDK provided by UserVoice

Chosen

Javascript library used to make ‘long, unwieldy select boxes much more user-friendly’. Used to improve user experience when categorising transactors, including search.

Licence - MIT license

Highcharts

Highcharts is a charting library written in HTML5/JavaScript. It’s used to draw the pie charts on the home page and charts throguhout the application.

Licence - Creative Commons Attribution-NonCommercial 3.0 License.

Base Admin

Admin theme, used on the non user facing admin pages.
License - Commercial Single Use license

Appendix E

PHP Code

Several pieces of functionality implemented in PHP were referenced throughout the report. Samples of code that were deemed too long or inappropriate to show in-line are listed here.

```php
/**
 * Checks for an existing transactor, creating a new one if not
 * @var $name The raw transactor string
 */
public function setTransactor($name) {
  // Trim off extra whitespace
  $name = trim($name);
  $this->setRawTransactor($name);

  // Tidy the name
  $name = preg_replace('/\s_/', '␣', $name);
  $name = preg_replace('/\s+/', '␣', $name);
  $name = preg_replace('#[^\w\s/\\-.]#', '', $name);
  $name = trim($name, '-./\␣');

  // If last 2+ characters are numbers
  if(preg_match('#(?:[A-z]|\s){2,}(?:[\s]+)?(\d{2,})$#', $name, $matches,
       PREG_OFFSET_CAPTURE)) {
    if($matches[1][0]) {
      $this->setNumber($matches[1][0]);
      $name = trim(substr($name, 0, $matches[1][1]));
    }
  }

  // Existing global transactor mapping?
  $globalTransactorMap = GlobalTransactorMappingQuery::
      findOneByTransactorName($name);
  $foundGlobalMap = (boolean) $globalTransactorMap;
  if($foundGlobalMap) {
    $this->setGlobalTransactorMapping($globalTransactorMap);
  }

  // Create a new user mapping and persist it
  $currentUser = $this->getUserSession()->getUser();
  $userMap = UserTransactorMappingQuery::findOneByTransactorNameAndUser(
      $name, $currentUser);

  if(empty($userMap) && !empty($name) && !$foundGlobalMap) {
    $userMap = new UserTransactorMapping();
    $userMap->setName($name);
    $userMap->setUser($currentUser);
    $userMap->save();
  }

  if($userMap) $this->setUserTransactorMapping($userMap);
}
```

Figure E.1: PHP `Transaction->setTransactor($name)` implementation

```
if($dmy && $mdy || !$dmy && !$mdy)
    // ... prompt the user
else
    // ... continue conversion using the detected month format
```

Figure E.2: Evaluating the results of the month format detection

```
public function getThrottlingWaitSeconds() {
  $throttleSteps = array(60 => 1, 120 => 2, 240 => 3);
  $throttlePeriodMinutes = $this->getConfig()->getThrottlePeriod();

  $mostRecent = FailedLoginAttemptPeer::GetMostRecent();
  if($mostRecent) {
    $mostRecentLoginTimestamp = $mostRecent->getUpdatedAt('U');
    $attemptCount = FailedLoginAttemptPeer::GetCountInLastMinutes(
        $throttlePeriodMinutes);

    // ensure the largest is considered first
    krsort($throttleSteps);
    foreach($throttleSteps as $attempts => $wait) {
      if($attemptCount > $attempts) {
        return time() - $mostRecentLoginTimestamp - $wait;
      }
    }
  }
  return 0;
}
```

Figure E.3: Calculating wait time following a failed login attempt

Appendix F

Suggestion Wizard API

The suggestion wizard is powered by AJAX, which makes JSON requests to a RESTful API on the backend. The figures in the section outline an example set of communication with the API, requesting the list of suggestions (Fig. F.4), mapping a reference to a transactor (Fig. F.2) and creating a new transactor (Fig. F.3), including an example response from the server following a POST request (Fig. F.1).

```
{
  "status": 0,
  "result": "Successfully mapped sainsburys to Sainsbury's (Global)."
}
```

Figure F.1: Response from API following a successful map or create

```
{
  "from": {
    "userMappingID": 443
  },
  "to": {
    "globalMappingID": 12
  }
}
```

Figure F.2: POST request sent to `/ajax/transactor/map`

```
{
  "name": "Tesco",
  "category": 34,
  "subcategory": null,
  "from": {
      "userMappingID": 464
  }
}
```

Figure F.3: POST request sent to `/ajax/transactor/create`

```
{"status":0,
   "result":[
       { "mapping":{
            "id":449,
            "name":"edf energy",
            "transactorName":"Edf Energy"
         },
         "exampleTransactions":[
            {   "id":1552,
               "value":"-169.00",
               "date":"2014-03-24",
               "memo":null
            }
            ... etc
         ],
         "suggestions": [
             "globalMappings":[
                {   "id":134,
                   "name":"e.on energy",
                   "transactorName":"E.ON"
                },
                {   "id":222,
                   "name":"edf energy-dom",
                   "transactorName":"EDF Energy"
                },
                ... etc
             ],
             "globalTransactors":[{
                   "id":149,
                   "name":"EDF Energy"
             }]
         ]
       },
       ... etc
   ]
}
```

Figure F.4: GET request sent to`/ajax/transactor/suggestions`

,

Appendix G

Questionnaire

Part 1

Answers on a 5 point scale, from strongly disagree to strongly agree. The questions were adapted from The Standardized Universal Percentile Rank Questionnaire [67].

1. I am able to find what I need quickly.
2. I enjoy using the website.
3. It is easy to navigate.
4. This website keeps the promises it makes to me.
5. I feel comfortable using the site.
6. I can count on the information provided on the site.
7. I feel confident uploading my statements to this website.
8. The information on this website is valuable.
9. If the website were completed, I would be likely to recommend this website to a friend or colleague.
10. If the website were completed, I would visit this site in the future.
11. I find the website to be attractive.
12. The website has a clean and simple presentation.

Part 2

Answers on a 7 point scale from very difficult to very easy, each question was paired with a free text field where users could provide additional information why. The questions were designed following the Single Ease Question format [68].
How easy or difficult did you find it to:

1. Download your statement from your bank? Why?
2. Upload your statement? Why?
3. View and understand your statement (on pegFinance)? Why?
4. View you monthly predictions? Why?
5. Overall, use the website? Why?

Part 3

The final part of the questionnaire was used to identify features of the the website that users favoured and to allow for feature requests, all questions were answered in free text fields.

1. What single feature of the website would you change?
2. Which feature(s) do you find the most useful?
3. What feature(s) would you like to see added?
4. Any other comments?