# Project Management

## Documentation of Problems:

This section is a brief introduction to some of the key problems faced throughout the project. Each point will be expanded on throughout the rest of the report, whenever necessary.

After organising the team for the project one member had dropped out, leaving the team as a four. As this happened quite early on the team was able to adjust and ensure we got the most out of the project. One issue that was difficult to resolve from this problem was communication. The team were not aware of the situation of the member. This situation was understandable, however had caused an issue.

During the initial planning stage of the project we were required to create an initial planning document. Even though this document did provide help to the team we were never given any feedback on it. This caused some confusion between the team as we felt that we had no confirmation on whether the direction of the project was correct. This was, however, later resolved in progress meetings with the tutors.

In relation to the organisation of the project the team felt the cognitive walkthrough and usability studies were set too early into the project. Even though the usability studies provided feedback with which we were able to improve our applications they would have been more useful towards the end of the project. The reasoning behind this is that for the first usability study the team had to organise a false representation of an application to be tested. This meant the results gained from the study were useful in relation to the design but not in relation to a developed application.

Due to other commitments and lack of awareness towards the end of the project the team had to crunch to ensure that the final report documentation was finished. Even though, in retrospect, enough time was allocated it felt like we were rushing to meet the deadline for this coursework and others. This problem was solved by delegating work evenly and ensuring that each member of the team had a clear idea of what was expected of them.

## Risk Analysis:

The initial risk analysis identified areas in which there was potential for risks to happen. The analysis also provided a guide to how probable the situation was, the severity of the situation and the overall risk factor. Included in the same table were measures to be taken in the case of the risk happening.

At the beginning of the project the team had lost a member, this risk was identified in the risk assessment. By following the controls set out in the risk assessment appropriate measures were taken; contacting the module leader and delegating work. The loss in a team member also resulted in a reflection on the realistic amount of work producible.

In relation to the Android application the team’s Android leader felt that inadequate online-help and documentation for Android Studio (Android IDE) was available. The official Android documentation was outdated in comparison to the IDE. A number of key methods and patterns were deprecated yet no new documentation was provided. Although the risk was identified in the risk analysis, the controls/action to be taken were not properly stated initially. Therefore the risk analysis provided no guidance on the realistic approach to take.

The risk analysis provided in the initial planning document was appropriate in relation to the project. Identifying these risks enabled the team to act accordingly and allowed for early adjustments to be made to mitigate from issues that may have arisen.

## Project Conduct:

## Pivotal Tracker:

The team made use of Pivotal Tracker as a tool for managing the project. Each user requirement was organised into stories which were added to the Project Backlog. These stories were then worked through in order of importance. Pivotal Tracker provided a visual representation of the work achieved, alongside the applications, as well as a representation of the amount of work still needed to be done.

Once the applications had been mostly developed, and only desirable stories remained, the use of Pivotal Tracker reduced. This reduction happened as team members had other commitments to attend to, core functionality was being worked on and small adjustments were being made to fine tune the applications.

Overall the use of Pivotal Tracker has been beneficial to the team and the project. The ability to keep record of who owned which story enabled the team communicate with each other in relation to specific functionality. This was extremely beneficial in producing two systems that related to each other in terms of layout, design and functionality.

## Bit-Bucket:

Bit-Bucket was a useful tool to use during the project as it provided the team with a way of managing code and being notified of any changes that had happened. Bit-Bucket also provided visibility of who had done work and the work they had done. Being able to review work, on the Bit-Bucket website, and seeing the changes through the commit logs enabled team members to see the changes that had been made and comment on whether the code could be improved. As well as providing version control Bit-Bucket also provided a level of security. As the code was stored on a repository the work was backed up, if a team member’s work had been corrupted or a major change had been made that they weren’t sure how to fix, a team member could just remove the code off their machine and replace it with the code of the repository. Although this was never the case it provided reassurance that a previous, working, application was available.

An issue with using Bit-Bucket was not having enough knowledge of how the repository worked. This was made apparent when team members were working on the same lines of code. This resulted in one member pushing all of their recently edited code and getting the other member to manually reflect on the code and add their changes to it. Although this was an issue it didn’t overly effect the state of the project but it did mean valuable time was spent reviewing on code. This could have been avoided through clear communication and a valuable lesson has been learnt. Another issue, although not greatly important, became apparent during pair-programming. Dependant on whose machine was being used meant that the Bit-Bucket commits only showed commits from one user. Upon reflection the team has realised we could have included the names of the members working on the specific work together in the commit logs.

Overall Bit-Bucket has been an invaluable tool to this project. The apparent issues with using Bit-Bucket do not reflect on Bit-Bucket directly but on our level of experience with it. Throughout this project the team has learnt a lot about the importance of version control and the clear advantages of using a version control system.

# SCRUM Framework:

The team members continued their roles in SCRUM throughout the project. During this time daily communication was organised to ensure that everyone was up to date with the progress of the project. If a member wasn’t available for face to face meetings due to other commitments another method of communication was organised, this was usually a group conversation on Outlook.

For SCRUM meetings minutes were recorded. These recorded minutes provided insight into the current progress of the project. Any issues brought up during the meetings were noted and discussed. The discussions lead to the team either finding a solution to an issue or providing their own opinion. Throughout some meetings potential changes to the design of an application or new features to implement were also discussed.

Although every element of the SCRUM framework and the agile methodology were not implemented key aspects of it were. These key aspects allowed the team to work together dynamically. During the initial planning stage this allowed us to organise better as a team. As the project went on other university commitments, deadlines and the Easter break broke up the fluid style of Agile. The team felt that Agile and SCRUM would be better suited to a project in industry rather than one at university. The team also feel that the lessons learnt from adopting a semi-agile process and the tutorials on the subject have improved our understanding of the methodologies available when working on a project.

# Desktop Application: Requirements Analysis and Design

## Class Design

In comparison to the initial plan for requirements analysis and design a number of changes have been made that reflect the growth of the application as well as critical areas that were overlooked.

The original class diagram (INCLUDE IN APPENDIX?) has changed during the course of the project, however the fundamental idea of how the classes should relate to each other remains the same. The initial class diagram planned on using an enumeration class for the *ItemType* and *AdvertType* classes. This changed however during implementation as it became apparent that if any changes to the information provided in the database was changed this would not be reflected in the enumeration class until the code had been edited. Instead it made more sense to create an *ItemTypeList* and *AdvertTypeList* that held these values. This meant as the database grew so would the application without any changes needed to the code. This approach was also used in the *ItemCategory* class. A mistake has been made in accordance to the relationship between these classes and an *Advert*. As each of these classes are related to the *Advert* they should have been included in reference to the *Advert*. Instead the GUI references the list as it populates each of these lists on initialisation. This is bad practise as the *Advert* does not contain an item type, advert type or category but instead contains pointers (ID’s) to the specific type in the related lists.

The review information was not implemented into the application until the final stages of development as the concept of how a review would work had not been completely agreed on. Therefore an independent *ReviewList* and *Review* has been included. This is again populated on initialisation and is referenced by the *Transaction* class. Upon reflection a *Review* should be an object directly relating to the *Transaction* as opposed to being simply referenced by an ID.

A feature added in to the application after design was the ability to add and edit rules. As the desktop is the only application this can happen in it was imperative it was included. As this class doesn’t interact with any other classes it is visualised in the class diagram as being independent.

A design change unique to the Desktop Application is the Admin Authorisation class. This class allows for the administrator to login to the system. The ability to be able to add new Administrators to the system has been implemented however this goal was not identified during the initial planning phase.

Due to time constraints these errors have not been fixed, however the application does still function as planned. Given more time the application would be reviewed and these changes would be implemented.

## Interface Changes

From the initial design document the desktop application has changed dramatically. This has been due to the feedback from the cognitive walkthrough and usability studies. Originally the application was going to be set out in individual frames, each offering the desired information dependant on the button the user has selected. Instead the application now provides a tabbed pane approach. This approach means the user now has the choice over workspace layout. For instance the user can view the system from the point of view that the member is the central point, this is called the Member Pane. Alternatively the user can use the Advert Pane which uses the individual adverts as the point of focus. This changed was developed from the initial cognitive walkthrough feedback. The tabbed pane view works well in allowing the admin to see the application holistically. This approach was considered useful in the usability studies as users felt they had a better overview of the system and didn’t have to remember information from another pane.

One disadvantage to this approach to layout is evident on the Transaction Pane. There is a lot of information in regards to Transactions and therefore the page is completely filled with information. To break up the information, to try and avoid information overload, each piece of information is grouped into titled panes.

## Missing Features

### Not included due to time restraints:

* Re-Activate membership or advert :

A user and an advert can be banned due to an offence (breaking a rule) or inactivity for an extended period of time. Once a member is banned they cannot currently have that revoked.

* Loading screen between login and loading the application :

After the administrator has logged in there is not a splash/loading screen between the login and the initialisation of the application.

* Contacting a member:

The ability to contact a member to notify them of any rules they have broken, notifying them of inactivity and other system information would have been useful. This goal was considered a stretch. Due to time constraints the application does not support this feature.

### Not included by design:

* Not included the API for the desktop application:

The desktop application did not use the API created for the Android and Web application due to the fact that the desktop application has separate functionality. The functionality needed for the desktop application is included within the application and therefore an API would not be necessary.

* Using Oracle Text to search through the application :

The desktop application searches through the table model so that once a key has been pressed the list is filtered. This differs from the Android and Web application, which uses Oracle Text. Oracle Text sends the searched word to the database where all the relevant columns have been stored to search. If the searched term matches a term in the searchable columns those columns are returned. This is beneficial to Android and the Web application as they are updated from frame to frame, however the desktop application updates dynamically when changes are made. This meant filtering the list was a better decision to make.

* Cannot create an administrator through the application :

For the desktop application there is already a pre-set administrator. This administrator was assigned to the database by the team members of the project. If a new administrator was required to join the system they would have to ask for permission to include this administrator in the database. The reason for this decision was security conscious. The team felt allowing anyone to register as an administrator to the application was bad practise.

# Desktop Application: Software Engineering

## Noteworthy Features

The desktop application has unique features that are not used in the Android or Web application. The reason for these unique features is that the desktop application is specifically created for administrators. Administrators require close access to information stored in the database in case any information is not correct, rules are being broken or an overview of the system is required.

The desktop application makes use of a recent Oracle implementation that allows for changes made in a database to be used to notify Java applications. The reason for implementing this technology is that it provides the administrators of the system to be notified of any changes made to data in the database. This notification then allows the administrator to have the option to update the information they are viewing. If the administrator does not update the application it will automatically update within 2 minutes. This has been put in place to ensure that the most up to date information is being viewed. An issue implementing this technology was that a specific port was required to be opened to allow the notifications to be sent through. This was an issue within university as permission to allowed ports to be opened is blocked by privileges.

Initially the application, whenever dealing with network tasks, was taking a long time to respond. The assumed reason for this was the fact that the GUI, Data model and Networking were all being done on the main thread. To speed up the application all networking tasks were split off onto another thread and ran, once the networking tasks were completed the GUI and data model would then be updated. This dramatically increased the speed of the application.

To ensure that administrators and user passwords are safe they are saved in the database as encrypted, hashed passwords. This was done with the use of a library called JBCrypt (Open source). By encrypting administrator passwords and user password has meant that the plain text version of their passwords are never stored. This has meant that when dealing with resetting passwords it has been easier to simply re-assign a new password than it is to send them their old password. Given more time a system for the users could have been put in place to send them an email on how to reset their email or containing a plain text version of their password. The team has decided that the current way of dealing with passwords is safer for security purposes.

For each of the fields that require administrator input regular expressions have been assigned to the input of the text fields. These text fields highlight green to hint that the input meets the correct format, if the input is red this would suggest the input is incorrect. If any of the text fields have incorrect information in them the administrator will be notified they cannot make any updates until this has been corrected. This front end data integrity is backed up by the database with back-end data integrity.

To increase the usability and fluidity of the application the Observer Pattern has been used. This design patterns allows the application to update all its information to all elements of the application being observed. The Observer Pattern means that when a change is made the information doesn’t have to be manually updated meaning the administrator is consistently up to date with current information.

## Problems and Issues

The desktop application planned to use a Singleton class to deal with a database connection. While developing the application it became apparent that this method was not necessary for this application. The ability to open and close the database connection has meant that networking is only used when necessary, this frees up resources. Other advantages of not using a Singleton class involve improvement to the flexibility of the class, a Singleton class is highly coupled across the whole application making it difficult to modify.

To provide the user with the ability to undo and redo changes the Command Pattern was supposed to be implemented. Due to time constraints this was left out of the application due to it’s complexity. However, in retrospect, this is a useful part of the system. Not including the Command Pattern does not directly affect the goal of the application; to provide the administrator with access to the database. Including the Command Pattern would of greatly improved usability within the application.

## Coding Practice

The desktop application has been developed with coding practices in mind. For instance the application reduces the amount of code re-use. This was achieved by splitting the GUI class into multiple classes, each with an individual purpose. The application was initially divided into separate packages but due to the package principles was developed into libraries. By splitting the classes into their individual packages means that the code can be re-used over a number of applications.

Where necessary Interface and Abstract classes have been used to improve the code quality. These class types improve code quality by reducing the amount of code repeated, forcing other classes to implement methods and improves code readability. To help improve the readability of the code the Java code conventions have been followed. Following Java code conventions has meant that the team has been able to understand each other’s development easier. Java documents were generated for the data model and comments were added where necessary to help understand the flow of code.