Panto Genie: A Theatre Show Production Tool

Jamie Holwill
H00173186

August 2014

Computer Science
School of Mathematical and Computer Sciences

Dissertation submitted as part of the requirements for the award of the degree of MSc in Information Technology (Software Systems)
Abstract

The planning, production and execution of a theatre show is a very complex endeavour. This Master’s project intends to aid the administrative burden for a director, through the production of a mobile application that automates administrative tasks and provide tools to effectively manage and direct a show.

The findings of requirements analysis show that there are many areas where a mobile application can help achieve the stated objectives and furthermore, provide functionality to other members of the theatre team. The main feature identified is machine understandable script integration allowing queries to be performed on the script for various purposes.

The report investigates the state of the mobile market and concludes that Android and iOS mobile devices should be targeted to ensure the application can reach the greatest number of users. The iPad is the development device due to the reduced fragmentation for testing.

The development section details all the processes carried out in producing the application. This includes details of specific technologies used and how code is structured to solve problems faced. The script document was successfully parsed into an XML format for integration with the application. Development was performed using Appcelerator’s Titanium framework and all planned features were implemented successfully.

Evaluation of the application was performed to assess the success of whether the project objectives have been realised. Evaluation included: a comparison of the delivered application to the intended requirements, the results of user testing and comparison of the application to other related works. The results of the evaluation suggest that the application met the base objective, but furthermore was considered highly usable and successfully met user needs in new and unique ways.

The report finally presents further work that could be carried out to enhance the developed application further in future iterations.
Statement of Non-Plagiarism

I, Jamie Holwill, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:

Jamie Holwill

Date: 21/8/2014
Contents
Abstract.............................................................................................................................................. i
Statement of Non-Plagiarism.................................................................................................................. ii
List of Figures ...................................................................................................................................... v
List of Tables ...................................................................................................................................... vi
1 Objectives........................................................................................................................................ 1
2 Literature Review ............................................................................................................................ 2
  2.1 Requirements Analysis .............................................................................................................. 2
    2.1.1 Users ...................................................................................................................................... 2
    2.1.2 Planning and Co-ordination .............................................................................................. 3
    2.1.3 Specific Responsibilities .................................................................................................. 4
    2.1.4 Stage Management Toolkit ............................................................................................. 10
    2.1.5 Networking ...................................................................................................................... 12
  2.2 Target Platform .......................................................................................................................... 13
    2.2.1 Platform Options ........................................................................................................... 13
    2.2.2 Difficulties of Mobile Platform Application Development ........................................ 13
    2.2.3 Operating Systems ......................................................................................................... 13
    2.2.4 Hybrid Development ..................................................................................................... 15
    2.2.5 Type of Mobile Device .................................................................................................. 16
    2.2.6 Target Device .................................................................................................................. 16
    2.2.7 Conclusion and Further Actions ..................................................................................... 17
  2.3 User Interface Design ................................................................................................................ 18
    2.3.1 Design Principles .......................................................................................................... 18
    2.3.2 Design Process .............................................................................................................. 19
  2.4 Storage ......................................................................................................................................... 20
    2.4.1 Relational versus Non-Relational Databases .................................................................. 20
    2.4.2 Features Requiring Storage .......................................................................................... 21
    2.4.3 Script Integration Storage ............................................................................................... 22
3 Methodology ..................................................................................................................................... 23
  3.1 Design and Build ....................................................................................................................... 23
  3.2 Testing .......................................................................................................................................... 23
    3.2.1 Basic requirements .......................................................................................................... 23
    3.2.2 Related work ................................................................................................................... 24
    3.2.3 User Testing .................................................................................................................... 24
  3.3 Planning and Risk analysis ........................................................................................................ 24
4 Requirements Analysis .................................................................................................................. 25
  4.1 System Requirements ............................................................................................................... 25
  4.2 Mandatory Requirements ....................................................................................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Optional Requirements..................................................................</td>
</tr>
<tr>
<td>5</td>
<td>Professional, legal, ethical, and social issues..........................</td>
</tr>
<tr>
<td>5.1</td>
<td>Professional issues ....................................................................</td>
</tr>
<tr>
<td>5.2</td>
<td>Legal Issues .............................................................................</td>
</tr>
<tr>
<td>5.3</td>
<td>Ethical Issues ..........................................................................</td>
</tr>
<tr>
<td>6</td>
<td>Development Process ....................................................................</td>
</tr>
<tr>
<td>6.1</td>
<td>Script Parsing...........................................................................</td>
</tr>
<tr>
<td>6.2</td>
<td>Titanium Framework and Development Process...............................</td>
</tr>
<tr>
<td>6.3</td>
<td>Script view................................................................................</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Part Highlights and Contents....................................................</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Comments....................................................................................</td>
</tr>
<tr>
<td>6.4</td>
<td>Checklists..................................................................................</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Adding New Items........................................................................</td>
</tr>
<tr>
<td>6.5</td>
<td>Contacts......................................................................................</td>
</tr>
<tr>
<td>6.6</td>
<td>Rehearsals..................................................................................</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Saving Rehearsals.......................................................................</td>
</tr>
<tr>
<td>6.7</td>
<td>Timer..........................................................................................</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation...................................................................................</td>
</tr>
<tr>
<td>7.1</td>
<td>Requirements Comparison.............................................................</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Couchbase Storage........................................................................</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Deliverables Status.....................................................................</td>
</tr>
<tr>
<td>7.2</td>
<td>Usability Testing.........................................................................</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Analysis of Results.....................................................................</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Conclusion on SUS results.........................................................</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 - Example script with blocking comments (Dean and Carra, 1989: pg. 281) ........................................ 5
Figure 2 - Rehearsal 2 electronic script and highlighting feature (Sotto Voce Film + Works, 2014a) ... 6
Figure 3 - Rehearsal 2 screenshot showing comment button (Sotto Voce Film + Works, 2014b) ........ 6
Figure 4 - Example ground plan including blocking (Dean and Carra, 1989: pg. 280) ......................... 7
Figure 5 - Theater Blocking screenshot showing basic blocking functionality (montysmagic, 2014).... 7
Figure 6 - StageWrite screenshot showing the advanced stage design and blocking tools (Open Jar Productions, 2014b) ........................................................................................................................................ 8
Figure 7 - Scene and Cast Breakdown example chart (Dean and Carra, 1989: pg. 341) ................. 9
Figure 8 - Breakdown of Scenes and Musical Numbers Example (Dean and Carra, 1989: pg. 342).... 10
Figure 9 - ShowTool SM screenshot showing the timer functionality (Murfin, 2014) ...................... 11
Figure 10 - ShowTool SM "Show Info" tab screenshot (Murfin, 2014) ............................................ 12
Figure 11 - Mobile Operating Systems 4Q13 Market Share by OS (International Data Corporation, 2014a) ........................................................................................................................................ 14
Figure 12 - Tablet Device Market Share by Developer (International Data Corporation, 2014b) ...... 17
Figure 13 - Incremental Life Cycle Model (ISTQB Exam Certification, 2013) ................................. 23
Figure 14 - Testing disclaimer shown to participants before testing ......................................................... 29
Figure 15 - Overview of the Panto Genie design .................................................................................. 32
Figure 16 - Panto Genie screenshot - Example of the Tab bar for accessing features .................. 32
Figure 17 - Representation of Script Page rendering process ............................................................. 33
Figure 18 - Panto Genie screenshot - Example of the contents picker for the script view to allow easy navigation ........................................................................................................................................ 34
Figure 19 - Panto Genie screenshot - Example of the script view with lines highlighted and comments ........................................................................................................................................ 35
Figure 20 - Panto Genie screenshot - Example of the checklist view showing both checked and unchecked items ........................................................................................................................................ 36
Figure 21 - Panto Genie screenshot - Example of custom input rows: first is standard text input and the second is a drop-down picker ......................................................................................... 36
Figure 22 - Panto Genie screenshot - Example of the contacts view with placeholder contacts and search bar ........................................................................................................................................ 37
Figure 23 - Panto Genie screenshot - Example of add new contacts screen with automatically populated drop-down picker for contacts ......................................................................................... 37
Figure 24 - Panto Genie screenshot - Example of native date picker being used with the custom drop-down picker ........................................................................................................................................ 38
Figure 25 - Panto Genie screenshot - Example of the pre-populated contact editing within pop-over ........................................................................................................................................ 39
Figure 26 - Panto Genie screenshot - Example of a rehearsal item in the main rehearsal view, categorised by time ........................................................................................................................................ 40
Figure 27 - Panto Genie screenshot - Example of the timers in the three states .................................. 41
Figure 28 - The SUS scoring system and grading (Bangor et al., 2009) .................................................... 44
Figure 29 - Bar Chart presenting SUS Score by User ..................................................................................... 46
Figure 30 - Signed Ethics Approval form .................................................................................................. 57
Figure 31 - Overview Gantt chart .............................................................................................................. 59
Figure 32 - Planning Phase Gantt chart ..................................................................................................... 60
Figure 33 - Development Phase Gantt chart..............................60
Figure 34 - Testing phase Gantt chart..............................61

List of Tables

Table 1 - Theatre Production Roles and Responsibilities (Dean and Carra, 1989) .................3
Table 2 - System Requirements .........................................................25
Table 3 - Mandatory Requirements .............................................26
Table 4 - Optional Requirements ..................................................27
Table 5 - Example conversion from plain text script line to XML marked-up version ....30
Table 6 - Status of the planned requirements ....................................43
Table 7 - Standard SUS questions (Brooke, 1996) ..................................45
Table 8 - Summarised Usability Testing Results ..................................46
Table 9 - Description of feature improvements and bugs noted from users’ testing debriefs ..........48
Table 10 - Suggested Future work, prioritised by customer needs .........................50
Table 11 - Risk Assessment ..............................................................62
Table 12 - User tasks for usability testing, including information on features tested ..........63
Table 13 - Individual and Aggregated SUS scores ................................65
Table 14 - Unadjusted SUS scores .................................................66
1 Objectives

The objective of this project is to produce an application to help in the planning, production and execution of theatre shows. Producing a theatre show is complex due to the level of co-ordination required between the director of the show and all the members of the cast and crew. The cornerstone of any theatrical production is the script; therefore the application needs to use the script as its central element. The application intends to reduce the level of administration burden for the director by providing tools that will automate actions and help organise the different teams in an efficient manner.

The project objective was chosen due to from experience of having directed and having been a member of the cast in amateur productions. Even at an amateur level, the level of co-ordination needed in producing a show is high. The availability of an application that would have assisted in coordinating the process would have undoubtedly resulted in better quality shows; time would have been released for further artistic interpretation of the script.

As part of the project research, the full requirements of theatre production teams have been assessed to understand the areas the project application should support. The tools primarily intended for the director will also benefit other members of the cast and crew. Having identified potential features, the report assesses the technical areas required for development of an application.

Development of the mobile application, based on the feature and technical requirements identified, will be undertaken. The development process, including any difficulties faced, will be discussed and documented.

Following development, the application will be evaluated to ascertain the level of success in meeting the objective and requirements. This will include user testing to assess the final application’s usability. Having assessed the success of the project, further work will be identified that would benefit the potential users of the application and improve upon the initial success.
2 Literature Review

When developing an application, planning and research is required to ensure that the end-product functions in an expected manner. The relevant topics for this project requiring further research are:

- Requirements Analysis
- Target Platform
- User Interface Design
- Storage

Each of these areas has been considered in greater detail in the following sections.

2.1 Requirements Analysis

One of the most important tasks in the software development process is confirming the user requirements during the planning stage. This allows for more efficient planning and will limit the number of changes that need to be made as the application development progresses.

The application is intended to function as an aid in the production and execution of a theatre show; this definition is relatively broad and will be refined to understand who the potential application users are and what functionality they may need.

Analysis will include reviewing related works to understand what features they offer and how they could be improved.

The full range of possible features will be identified in this section. However, the final decision over the features to be included in the final project will be carried out in Section 4.

2.1.1 Users

The main intended user of the application is the director of a production. One of the director’s main roles is to organise the entire production of a play: they must analyse the script from an artistic point of view, applying their knowledge of theatre to create a show and they must organise the entire process from rehearsal to production (Dean and Carra, 1989). The director may therefore need tools that range from helping with the analysis of the script right through to the co-ordination of his team.

The team that the director is required to coordinate also present the majority of the different possible users for the project application. According to Dean and Carra (1989), there could be as many as 13 different roles involved in a production, each with their own distinct responsibilities. The number of roles increases by four in musical theatre, where further consideration for the dancing and music must also be taken into account (Dean and Carra, 1989). Each member of the team has distinct requirements that could be addressed by the project application. The main team members and relevant responsibilities, have been summarised in Table 1 (Dean and Carra, 1989).

<table>
<thead>
<tr>
<th>Roles</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cast</td>
<td>• Memorising lines and stage directions provided by the director</td>
</tr>
<tr>
<td></td>
<td>• Attending rehearsals organised by the director</td>
</tr>
<tr>
<td>Stage Manager</td>
<td>• Assists the director in casting</td>
</tr>
<tr>
<td></td>
<td>• Gets details of all cast and crew</td>
</tr>
<tr>
<td></td>
<td>• Checks with Equity to ensure if staff are members</td>
</tr>
<tr>
<td></td>
<td>• Arranges the rehearsal space</td>
</tr>
<tr>
<td></td>
<td>• Lay out plans of set on floor for crew</td>
</tr>
<tr>
<td></td>
<td>• Arrange rehearsals props and furniture</td>
</tr>
<tr>
<td></td>
<td>• Posting all notices, calls for rehearsals</td>
</tr>
<tr>
<td></td>
<td>• Record rehearsal time</td>
</tr>
</tbody>
</table>
• Organises costume fittings
• Assign dressing rooms
• Holding the book (see section 2.1.3.1 below)
• Provide reports to producer and director of shows

Designers:
- Scene
- Costume
- Lighting
• Report to the director with designs
• Co-ordinating with the director and other designers to understand the artistic direction of the production
• Design all aspects of the production
• Supervision of the design execution

Technical Director
• Reports to the designers and director to co-ordinate his teams in the production of designs

Property Master
• Reports to technical director
• Designs and builds all the props
• Responsible for storage of all props once the shows are being performed

Music Director
• Co-ordinate with the director on their production intentions
• Rehearse the songs with cast
• Conducts the orchestra

Choreographer
• Co-ordinate with the director on their production intentions
• Blocks the dances, organising all dancing members of the cast

Table 1 - Theatre Production Roles and Responsibilities (Dean and Carra, 1989)

From examination of the different roles of the team members, a common theme becomes clear; many of the responsibilities of the director are delegated to sub-managers who specialise in their area of design or direction. Designs are delegated further to the technical director who co-ordinates the actual production work performed by a member of their crew.

The stage manager is an integral part of the production team and many features that could be implemented would help fulfil their responsibilities. They perform much of the co-ordination of the organisational tasks, leaving the director with more time to concentrate on his artistic vision.

The complexity of the entire production process is increased further by the deadline imposed by the first performance; this performance deadline is completely inflexible, which means all work has to be completed on time.

Based on the assessment of the different responsibilities, two types of features will be considered: features to aid in the planning and co-ordination of the different teams, and; tools to aid specific role responsibilities, encouraging timely completion.

2.1.2 Planning and Co-ordination

There are multiple layers of management when producing a theatre show. Therefore, the co-ordination of these teams needs to be as simple as possible.

Although a theatre production is primarily creative, unless there is a well thought out and carefully executed rehearsal schedule then the director will struggle to output a show (Dean and Carra, 1989). During the main rehearsal period, the director must be careful to stick to the schedule otherwise they will run the risk of over-rehearsing the earlier portions of the play and struggling to fully focus on the latter portions (Dean and Carra, 1989).

As with any project management exercise, more efficient planning and coordination will result in a superior final output. With less time spent carrying out administrative tasks, team members can properly focus on their responsibilities, with better quality result (Gupta, 2013).
2.1.2.1 **Timetabling**

Whilst no two productions are identical in the way they are planned and produced, there are some stages likely to be common between them all. The average production is likely to require a rehearsal schedule somewhere between four and six weeks in length. Within this rehearsal period, to have an effective performance, there should be at least one technical and three dress rehearsals (Dean and Carra, 1989).

There are a number of ways the application could help co-ordinate the rehearsal schedule. The application should have some form of calendar integration, with the option to invite specific members or groups within the team to rehearsals or meetings. The director could use this feature to plan out the rehearsal schedule and simultaneously send out the invitations to the cast and crew as applicable. Being able to group the different members of the cast and crew into different groups would be an invaluable way of efficiently posting calls for rehearsals.

2.1.2.2 **State of Deliverables**

The administration of delegating and supervising teams could be reduced by having an automated checklist of deliverables. This could be set at the beginning of a period of production and assigned to the relevant party to deliver on. As the different deliverables are met the status of these could be updated for the director’s review.

This automation could allow the director to take an overarching view of how the production is progressing without having to specifically request the information. By continually being able to check on the progress of various deliverables as production progresses, the director is less likely to have unexpected issues arising at the point of the technical rehearsal (Dean and Carra, 1989).

2.1.3 **Specific Responsibilities**

As well as the co-ordination tasks, there are many other responsibilities of the director and other members of their teams that must be fulfilled to ensure a successful production. Many of these responsibilities could be incorporated into the application and in doing so would enable them to be carried out more efficiently. These specific responsibilities have been considered in more detail in the sections following.

2.1.3.1 **Script Integration**

One of the main features that the final application should include is script integration. The script is the main resource that all users will refer to in their responsibilities. Many of the users would benefit from script integration as they would be able to identify all the relevant lines, cues, props within the script and take any notes easily.

Being able to annotate a script is an essential part of the production process at many stages and for many people: the director will take notes when analysing a play to record ideas on how best to present the scene, cast will record notes on their blocking and any performance notes, and the stage manager will use an annotated script to record all the different cues he will be responsible for and will annotate the script whilst holding the book (Dean and Carra, 1989).

Holding the book is a task carried out by the stage manager during the initial phases of the rehearsal schedule. Whilst the show is being rehearsed, it is the stage manager’s responsibility to take a note of: all lines amended, cut or added in the script; any actions or blocking provided by the director to the cast, and; small ground plans for the positioning of cast on stage (Dean and Carra, 1989). An example of an annotated script has been shown in Figure 1 (Dean and Carra, 1989).
As well as providing an option to annotate the script, further features around being able to search and highlight the script would be beneficial. Due to the flexibility of the electronic format, it should be possible to manipulate the script for many different purposes. For example, if members of the cast wanted to separately view, or even highlight all their lines they should be able to perform a simple query to filter the script for directions and lines pertaining only to their part.

A related work, Rehearsal 2, provides an electronic format for viewing and interacting with scripts. Restricted to Apple iOS devices only, Rehearsal 2 provides the user with the ability to: take notes, highlight lines and split the script into scenes for easier navigation (Sotto Voce Film + Works, 2014b). As the documents are word or PDF based (Sotto Voce Film + Works, 2014b), there are no advanced search features available as noted in the paragraph above. This is a clear area where further features could be implemented and the concept improved. Figure 2 (Sotto Voce Film + Works, 2014a) shows an example screenshot of the electronic script being highlighted on a screen. The highlighting shown in Figure 2 (Sotto Voce Film + Works, 2014a) has been done manually. To do this for an entire script could be quite a very lengthy task depending on the number of lines.
Figure 3 (Sotto Voce Film + Works, 2014b) shows an example of the electronic script in Rehearsal 2 that has been annotated by a user. A comment is hidden and the user must click on it to see what is written. Dean and Carra’s (1989) marked up script example, as shown in Figure 1, would be easier to follow given the script and comments can be seen simultaneously. If implementing the feature, to provide a better user experience which is as close to original scripts as possible, it should be possible to see the script and notes simultaneously.

2.1.3.2 Blocking diagrams

One of the first stages in the planning process, considered essential, is the creation of a ground plan (Dean and Carra, 1989). At its most basic level, a ground plan is a top-down view of the set including all the furniture, stage props and entrances/exits. It is a product of the scene designer, following initial discussions with the director and other designers and translates the director’s vision into a viable design (Dean and Carra, 1989).

The director uses the ground plan to document some of the blocking ideas. Blocking is simply the arrangement of the cast whilst they are on the stage. Figure 4 shows an example ground plan including cast blocking (Dean and Carra, 1989). This example demonstrates how complex both the ground plan and blocking could be for a scene; the diagram can be used to demonstrate the movements to the cast and crew, who will also mark up their scripts accordingly.
There are two examples of related works found that provide this functionality to users: StageWrite (Open Jar Productions, 2014a) and Theater [sic] Blocking (montysmagic, 2014). StageWrite provides a much high level of functionality compared to Theater Blocking; however the underlying concept of each application is very similar; they provide a platform for the user to mark up blocking on top of a ground plan.

Theater Blocking provides a number of template stage designs that the user can choose from, and allows the user to draw in the lines to reflect the intended blocking. Figure 5 provides an example of the application in use (montysmagic, 2014). The concept is simple when compared to the example in Figure 4, however it provides the basic functionality the user may require. A major drawback is that the ground plans will not be relevant for all productions, and will not include all details required, such as the placement of furniture.

StageWrite provides many more features compared to Theater Blocking. StageWrite allows a user to: design the ground plan in application, import the designers ground plan, advanced blocking design options and in-app sharing options (Open Jar Productions, 2014b). Figure 6 (Open Jar
Productions, 2014b) shows an example of the application in use. With comparison to Theater Blocking, the end result appears cleaner and is more flexible given the ability to import any stage design.

Figure 6 - StageWrite screenshot showing the advanced stage design and blocking tools (Open Jar Productions, 2014b)

The advanced features of StageWrite mean that the application is far more useful as a tool, but does come at a premium cost £139.99 (Open Jar Productions, 2014b) compared to Theater Blocking’s £0.92 (montysmagic, 2014). This difference in price reflects and helps explain the differing quality of the applications.

2.1.3.3 Charts
When producing a show such as a musical, there are many more teams involved. For this reason, there needs to be extra steps involved in the planning to co-ordinate all the different aspects: “dialogue... songs, dances, routines and musical accompaniment” (Dean and Carra, 1989: pg. 334). To co-ordinate all the areas effectively, Dean and Carra suggest the use of charts in the planning stage (Dean and Carra, 1989):

- Scene and Cast Breakdown chart – this is used to identify all the different roles and characters in each of the scenes. The breakdown can usefully record details of: costumes for different characters, songs in scenes and the level of involvement support cast have in the scene (singing, dancing or dialogue). Figure 7 shows an example of the chart as suggested by Dean and Carra (1989).
• Breakdown of Scenes and Musical Numbers – this specifically breaks down each scene by dialogue or musical and dance numbers. Once broken down, it identifies each of the participating characters and cast. This can be specifically used to enable the rehearsal of different production numbers independently and quickly identifies the cast who need to be called to the rehearsal. See Figure 8 (Dean and Carra, 1989) for an example.
If the application included charts, as discussed above, this could usefully be integrated with the scheduling function. As well as identifying all the scenes that would need rehearsals, the breakdown of cast members who need to be called would be easily identified.

2.1.4 Stage Management Toolkit

The stage manager is “in complete charge of the backstage” (Dean and Carra, 1989: pg. 348) and is therefore required to have knowledge of every aspect of the production. As a result of this role, the stage manager has some of the most varied responsibilities which could be incorporated into the project application. Dean and Carra (1989) stress the full list of responsibilities is likely to evolve under different circumstances, however the key tasks identified in Table 1 are relatively complete.

Some of the stage manager’s tasks, such as script annotation, have already been identified in the previous sections. However, a number of tasks could beneficially be included in the application.

2.1.4.1 Cast and Crew Information

While assisting in casting and keeping details of the cast and crew, a centralised database including all personal information should be kept. This could be used within the communication features too, as the application could automatically fetch user emails to send details of the rehearsal schedule.

2.1.4.2 Timings

Another important aspect of the stage manager’s role is the recording of rehearsal and performance timings. During the rehearsals the timing is used to get some indication as to how long the performance will be expected to last. During performances, the time for each scene and act is recorded to understand if the performance is taking longer than expected (Dean and Carra, 1989).
The related work, ShowTool SM, offers the timer functionality for its users (Murfin, 2014). Users are able to create different templates for the show timings and then record the timing of the shows. An example screenshot of this feature has been included in Figure 9 (Murfin, 2014). This shows the different timings for each of the different parts of the show. ShowTool SM also includes email support to allow for the easy transfer of the information recorded, this can then be used to create various reports (Murfin, 2014). Providing a report on the performance, including areas such as the show timings, is one of the responsibilities of the stage manager (Dean and Carra, 1989).

2.1.4.3 Checklists

It is important to ensure that the backstage management is as efficiently organised as possible. A well run backstage will ensure a smooth transition between scenes and will boosts confidence in the actors (Dean and Carra, 1989). One of the simplest methods to ensure an efficient management is through the use of checklists. Checklists ensure a methodical approach to each of the tasks that must be performed.

It is possible to find many examples of checklists for a stage manager during a show. Both Dean and Carra (1989) and Colby Theater and Dance (2012) have suggestions for pre and post show checklists. The checklist within the final application could use many of the same ideas for the types of items. However, to further enhance the feature, the checklists should be editable as applicable.

The related work, ShowTool SM, also provides a checklist feature, but does not appear to provide any template checklists (Murfin, 2014). One useful feature, shown in Figure 10, is the “Show Info” overview tab which shows the state of checklists in one place (Murfin, 2014). A similar feature could be incorporated into the final project application allowing for an overview of any and all checklists.
2.1.5 Networking

The feature which underlies many of the features above is how the application will communicate with other devices. Features, such as the rehearsal scheduling, require that the application can communicate with other members of the production team and cast to ensure that they receive the invites to rehearsals. However it could be restrictive, and detract from the practicality of the feature, if the application required that other users have the application to receive these invites. This feature should be met through the integration of the existing email on a device.

Certain features, such as the blocking diagrams or the state of deliverables, would benefit further if changes could automatically be pushed to other users on different devices. For example, the director could plan blocking and then share the diagrams with the cast to study on their own devices before a rehearsal. To properly use this feature, it would require a central-server based copy of data which could be synchronised with devices. This communication with the network, and the storage aspects involved, has been considered in section 2.4.
2.2 Target Platform

2.2.1 Platform Options
When developing this application, there are two main choices for the target device: the PC (for the sake of this thesis, this is considered to encompass Desktop PCs and laptops) or a mobile platform. Due to the nature of the application, it has been concluded the most logical device for development would be the mobile platform; the tool is intended to be used during rehearsals and production, and therefore the platform should be mobile to enable this style of use.

The mobile platform is a very popular platform; there has been considerable growth as evidenced by the sales of smartphones actually exceeding those of PC’s in late 2010 (Ohrt and Turau, 2012) and the total number of mobile devices reaching a total of seven billion in 2013 (Cisco, 2014).

The expectation is that this popularity will only increase further. Predictions state that the total global mobile data traffic is predicted to reach 11.2 exabytes\(^1\) per month in 2017 (Nicolaou, 2013). By comparison, actual global mobile data traffic had reached 1.5 exabytes per month by the end of 2013, this in itself was an 81% growth from the same time in 2012 (Cisco, 2014).

2.2.2 Difficulties of Mobile Platform Application Development
Whilst the mobile platform has been chosen for this application, the type of mobile device that will be targeted still needs considered. Developing applications for mobile platforms is not a simple process due to a high level of fragmentation in the types of devices available (OpenSignal, 2013). Developing for every platform could mean debugging as many as 20 different devices (Nithiyانantham and Kirubakaran, 2013). To reduce the level of testing, one device could be specifically targeted. However, if the device was not popular, the success of the application could be severely limited and a significant amount of effort may be needed to port the application across to other devices. The main areas that cause device fragmentation are (Nithiyانantham and Kirubakaran, 2013):

- Different Operating Systems (OS)
- Device shape and size
- Screen resolution
- Processor power

2.2.3 Operating Systems
There are several different OSs across the different mobile devices. The distribution of operating systems is shown in Figure 11 (International Data Corporation, 2014a). The OS with the predominant market share is Android with 78%, followed by iOS with 17% market share (International Data Corporation, 2014a). There is a clear gulf between Android and iOS and the other operating systems. For this reason, it has been concluded that the decision of which OS to target will be restricted to these two.

---

\(^1\) One exabyte is equal to 10 million terabytes
In choosing the target OS, an important aspect that must be considered is the type of application being created. There are three types of mobile application, two of which would support cross-platform development:

- **Native applications** are written directly in the platform owner’s Software Development Toolkits (SDKs) and are written in a specific development language. This is the most restrictive type of application in terms of cross-platform support, however is likely to be the most responsive as it can be fully optimised for the platform (Popa, 2013). Furthermore, these applications are able to access any of the features provided by the platform developer (Popa, 2013).

- **Web applications** are developed using web technologies such as HTML5, JavaScript and CSS3. The web pages are optimised for the mobile screen to have a similar appearance to that of a native application. The advantage of a web application is that all mobile devices can access them through their web browsers and so only the code for the website need be maintained (Popa, 2013). However, as the application is hosted on the web it is more difficult to access the device or platform features and the user must be online at all times to use the application (Popa, 2013) (Ohrt and Turau, 2012).

- **Hybrid applications** are “web application[s] wrapped in a native shell” (Popa, 2013: pg. 41). The application is developed using a platform independent language (such as JavaScript) and a virtual machine is required to interpret this code for the native development language (Ohrt and Turau, 2012). Hybrid applications combine the advantages of the native and web applications: they allow the developer to maintain a single codebase across all platforms, they allow access to the device’s features, and the device need not be connected to a network to access the application (Popa, 2013). The main disadvantage is that the performance of a Hybrid application will generally suffer from the requirement of running a virtual machine (Ohrt and Turau, 2012). However, with current mobile technology Charland and Leroux (2011: pg. 49) argue that this performance hit will be “negligible... in a well-built business application”.

Having assessed the different types of applications, it would appear that the decision must be made between a hybrid and native applications. Web applications have not been considered as it would be an unnecessary restriction to force the user to always need a network connection for the application.

![Mobile Operating Systems 4Q13 Market Share](image)
to function. Furthermore, the loss of access to device features (e.g. calendar and email) would create barriers for the planned requirements.

A hybrid application has been considered the most logical choice; by using hybrid technologies both iOS and Android mobile device owners would have access to the application. Per Figure 11 (International Data Corporation, 2014a), this would allow the application to target 95% of devices.

It should be noted that despite only needing to maintain one core codebase, the user interface (UI) design would require further work with each OS developed for. Each of the OSs have different conventions that result in UIs needing to be adapted (Charland and Leroux, 2011). For example, Android devices include hardware buttons for “back” and “menu” commands where iOS devices do not. Applications on the iOS platform therefore require that the UI include these options in their design (Charland and Leroux, 2011). The UI will need to be considered separately for each platform the application is published on which will result in some loss of shared code. This has been considered as part of Section 2.3.

2.2.4 Hybrid Development
Ohrt and Turau (2012) and Popa (2013) performed extensive comparisons of the different tools available to aid developers in the creation of cross-platform applications. However it should be noted, due to the rapid development in this area, these reviews have swiftly become unreliable. For example, Ohrt and Turau’s (2012) paper has considered version 1.7.1 of Appcelerator’s Titanium toolset; at the date of writing the current version is 3.2.2 (Appcelerator, 2014e). Despite this, it is clear from their analysis that no solution tested was clearly better than any others. To select the development tool to be used, the features of each should be considered to understand which would be most appropriate.

To help reduce the number of tools needing considered they will be restricted by their support for the Couchbase database system. See Section 2.4 of this report where the inclusion of the Couchbase platform has been discussed. Per Couchbase (2014b), the tools can be narrowed down to: Appcelerator’s Titanium, PhoneGap and Xamarin.

Xamarin will be excluded from consideration as this is a primarily paid-for service. The toolset has a free starter edition, but as applications grow in size the toolset needs paid for (Xamarin, 2014). The other development tools are free and open source and therefore suit the nature of the academic project (Appcelerator, 2014d) (Adobe Systems Inc, 2014).

The Titanium and PhoneGap frameworks use JavaScript as the main development language and both produce an application that runs on the target platform as a natively wrapped application (Whinnery, 2013). Where the applications differ is on their approach to the UI; Titanium requires that the user develop platform specific UI code, whereas PhoneGap utilises the phone web view and static web pages adapted for mobile (Whinnery, 2013).

The Titanium approach results in a more accomplished UI that takes advantage of the platform’s native properties, but this requires extra code to be written for every target device. Conversely, PhoneGap’s approach means that almost 100% of the written code will be reusable across many devices. However this reusability comes at a price as access to platform specific UI features will be lost (Whinnery, 2013).

Based on the differing approaches as to how the UI is designed, it has been concluded that Appcelerator’s Titanium toolkit will be used. The final application would result in a far richer user experience by developing using the native UI. This will result in more code needing to be written for every device that is supported, but this is deemed a suitable cost given the improved user experience.
A further benefit of using the Titanium toolkit is that there is well established backend support for network communications (Appcelerator, 2014b). One of the underlying features identified in section 2.1 on requirements analysis is that networking would be required to between different users and devices.

2.2.5 Type of Mobile Device
Having decided upon a hybrid application, careful consideration must now be made over the type of device that will be developed for given the short development time available.

As noted above, as well as the choice of OS, there is much fragmentation of the mobile platform due to: device shape and size, resolution and processor power (Nithiyanantham and Kirubakaran, 2013). This fragmentation is most significant across the range of Android Devices; at July 2013 there were 11,868 distinct devices noted, whereas there were just over a dozen different devices using iOS (OpenSignal, 2013). Whilst many of these different Android devices will be similar in their specifications, there will still be many different combinations requiring testing. Due to the development time available, it is considered out with the scope of this project to produce an application that will work across the majority of these devices.

To help narrow down the type of device that will be targeted for development, the planned features will be considered against the potential user experience. One of the main features affecting the user experience is the script integration. As noted in Section 2.1, being able to see any annotations at the same time as the script will provide the best user experience when considered against normal script usage. To achieve this experience, a bigger screen will need to be used.

Based on this, the ideal target device would be a tablet. The increased screen size would better support simultaneous viewing of script and notes. A smaller device could be potentially be used with a reduced user experience, however, this would need to be considered for further work due to the extra time constraints of designing the separate UI.

2.2.6 Target Device
With the development time available, this project will focus on developing the application fully for one device with possibilities for further expansion as future work. Figure 12 (International Data Corporation, 2014b) shows the highest single share of the tablet market belongs to Apple and therefore iOS. Though the overall majority of tablets will be Android the fragmentation of devices will be high.

Based on the market share data and the amount of fragmentation across the Android Platform, one of the Apple iPad devices is likely to be the most common singular tablet device in the market. Therefore the Apple iPad, with Retina displays, will be chosen as the development platform. This will allow the application to work across the greatest number of devices, as iPads face the least amount of fragmentation.
2.2.7 Conclusion and Further Actions

Given the research into the target platform and device, it has been concluded that the ideal solution will be to develop a cross-platform application, initially targeted at tablet mobile devices. The iPad has been chosen as the initial development platform as iOS has the least fragmentation when compared to Android.

The hybrid application requires that a specific development toolset will be required. From the research performed, the framework that has been deemed most suitable for the application is Appcelerator’s Titanium. This framework provides the ability to access native UI features and has a robust back-end support for networking the application.

Further suggested work would be: developing a UI for the Android platform, and exploring the possibility of porting the application to work on smaller devices such as smartphones.
2.3 User Interface Design
Getting the UI design right can have a large impact on the overall user experience. Users expect the application to deliver on promised functionality, but their opinion is just as likely to be influenced by the application’s UI (Apple Inc., 2014a).

The most comprehensive resource for designing a UI on the iOS platform, i.e. the target platform, is the “iOS Human Interface Guidelines” from Apple (Apple Inc., 2014a). Google have an equivalent set of guidelines for producing Android applications (Google Inc., 2014a) on their developer webpages. The guidelines provide advice on how to design the UI to give the best possible user experience.

It is not the intention that this section conclude on a complete UI design for an iOS tablet device. The intention is to: assess common advice for designing an application from both Apple and Google, the process that should be followed in designing an UI, and to understand whether any special considerations need be taken at the design stage.

2.3.1 Design Principles
The basic design principles for designing an application will be considered for both iOS and Android. These will be compared and contrasted to understand if there are any principles of one platform that contradict one another.

2.3.1.1 iOS
The iOS Human Interface Guidelines suggest the following six principles (Apple Inc., 2014a):

- **Aesthetic Integrity** – the design style should follow the same overall message of the application.
- **Consistency** – the application design should have consistent styles compared to what users are familiar with. This includes both other applications and internally.
- **Direct Manipulation** – don’t have separate controls for interacting with items on the screen.
- **Feedback** – provide confirmation when a user interacts with the application and provide progress information if the action will require any waiting time.
- **Metaphors** – if a real world object is represented virtually, have the interaction be as similar as possible.
- **User Control** – if an action needs to be started by an application, have the user confirm and approve this. The feeling of security and control will lead to a higher user satisfaction.

2.3.1.2 Android
Google’s design principles are not as succinct as Apple’s, however they are split into three categories of goals, the “Creative Vision”, which convey the main ideas (Google Inc., 2014b):

- “Enchant me” – the application must provide more than just attractive look. They must also feel good to use by being fast and smartly designed.
- “Simplify my Life” - the application must be intuitive to use, remove any tedious tasks from the concern of the user and the user should feel like they are in control. Easy tasks should be achieved through easy interaction and complex tasks should be intuitive to solve.
- “Make me amazing” – applications should make the user feel like they are able to use them in many different ways: “combine applications into new workflows through multitasking, notifications, and sharing across apps” (Google Inc., 2014b)

2.3.1.3 Comparison
Both the iOS and Android guidance documents promote applications to be intuitive and simple to use. There are different principles proposed by both Apple and Google, however they are not in direct opposition to the other
When programming the UI, there will be a separate process for each of the platforms as the UI code and specific features offered are different. The project application UI will be designed with the iOS guidelines, as the target device is the Apple iPad. As noted through the comparison of guidelines, by following the iOS principles the overall design should not require changing if ported to Android.

2.3.2 Design Process
Apple provide guidance on how best to design an application, including the UI, from the point of having an idea to the finished product stage; the recommended approach is (Apple Inc., 2014b):

1. Define the application requirements
2. Design the interface to meet the requirements
3. “Prototype and Iterate” (Apple Inc., 2014b)

The first recommended stage is to plan the application features with a focus on the end user. The features for the application have already been considered within section 2.1 and are refined further in section 4.

The second step requires that any custom interface designs should clearly meet the requirements of the application. With specific reference to the project application, this means that features such as the script integration must be designed in such a way that the views and different features offered did not detract from the experience of using an electronic version of the script.

The third stage has been considered as part of the development process; see Section 3.1 for further information.
2.4 Storage
The application features will require persistent storage between sessions. This section will assess the different methods of storing information.

There are two types of databases that can be used to store information:

- **Relational Databases**, such as SQL, is a structured way of storing information in tables that emphasises the relationships between information (Oracle, 2014).
- **Non-Relational Databases**, such as NoSQL, were designed to tackle the issues that arise from the relational database structure: rigid schema requirements, scalability and cloud capabilities. Information is stored in semi-structured JSON objects; this may create duplication of data, but is considered an acceptable approach due to the cheap cost of storage (Couchbase, 2014c).

The two types of databases will be considered, to understand which will provide better functionality for the project application. Following on from this discussion, the features that require database support have been identified.

2.4.1 Relational versus Non-Relational Databases
There are various arguments as to which of the database technologies should be used for mobile applications, however neither option is irrefutably correct and the benefits of each technology must be considered in the context of the application (Jatana et al., 2012). For example, Selvadurai (2012) promotes the use of NoSQL in cases where there are large amounts of text data held on a central server; traditional relational database management systems were found to be slower.

The three main arguments for using a non-relational database are: the flexible schema, scalability and cloud capabilities. These three areas will be considered in reference to the nature of the application.

2.4.1.1 Flexible schema
One of the main issues noted with relational databases is the rigid schema that must be applied to all relations in the database. In the case of very large datasets, updating the schema and information in the database will require a lot of effort and result in much downtime (Couchbase, 2014c).

Individual users of the project application, or even productions teams, are not likely to have a high volume of data. The intended features may have many different fields of data, but the overall potential size will be constrained by the theatre production; there is not likely to be more than three acts worth of data.

However, the target user base could potentially end up being very large in size. If the features such as script integration were to prove popular with cast, as well as management, then this increases the number of potential users to all the actors, directors and stage managers. If this were the case and changes needed to be made to scripts or user data schema, the application could potentially benefit from having a flexible schema to reduce any potential downtime in the event of changes.

Another benefit of having a flexible schema is the option for future application features to be implemented without needing to change, or be adapted to, existing schemas. As further work, features are likely to be added onto the project; the design goal is to provide a system which allows for new with as small an impact on existing code as possible.

2.4.1.2 Scalability
Relational databases are not inherently scalable across the millions of potential users that could access them across a network (Couchbase, 2014c). Issues with relational database structures mean that to scale up operations, a bigger and faster server needs to be installed to preserve data
integrity. Non-relational databases can be scaled up more easily through using multiple servers and distributing the database across these; this distributed approach is more cost effective and efficient (Couchbase, 2014c).

This particular issue with relational databases is not likely to affect the project application. The maximum number of users that would require access to a single database will be the number of people involved with a production.

2.4.1.3 Cloud capabilities
Due to the distributed server technology behind scaled non-relational databases, they inherently work better than relational databases with the distributed nature of cloud technologies. However, the cloud capabilities argument for non-relational databases will not impact the project application. The nature of the application has required that the features be available, which is not possible using traditional cloud computing.

However, non-relational databases are a better fit for databases that must be replicated to a number of different locations (Jatana et al., 2012). This is one of the potential features being considered for the application; to provide synchronisation between different users. The use of a non-relational database would provide a better foundation for development, by providing better support for the network features.

2.4.1.4 Conclusion for Database Type
The different types of databases have been considered and it has been concluded that, although there is no clear front runner, the non-relational databases offer the best features for the application. The main benefits that using a non-relational database will provide are: the flexible schema provides an easier basis for expanding the application with further features as it progresses, and better networking options for multiple users.

From analysis of the different technologies available, the NoSQL mobile database from Couchbase has been considered a suitable option for the project application (Couchbase, 2014b). The mobile version of their server is deemed appropriate for the application as it: offers local support on devices; it has simple synchronisation features, and; it has supported plug-ins for the hybrid development programme, Appcelerator’s Titanium, which has been chosen as the development tool for the programme.

2.4.2 Features Requiring Storage
Given the requirements considered in section 2.1, almost all the features will require persistent storage between sessions. The following features would require database storage:

- Rehearsal schedules
- State of deliverables
- Charts
- Cast and crew personal details
- Timing
- Checklists
- Blocking Diagrams – note, the data from these diagrams can be stored as serialised data. This is another benefit from choosing the non-relational database format where these different types of data are supported (Jatana et al., 2012).

Each of the features above will be easily presented as sets of data. The UI will need to provide an interface to view the data, performing background queries based on the user’s view and parameters.
2.4.3 Script Integration Storage

There are two approaches that could be taken for the script storage: it could be stored as a PDF document, or it could be stored as a marked up XML file.

The simple approach would be to store the script as a PDF file. The majority of features, such as annotation and comments, would still be supportable. However, PDF files are not machine understandable and therefore features such as querying the script and thus identifying all cues or lines for certain characters would not be supported.

If a script were to be stored as a marked up XML file, it would be possible for these further features to be implemented. XML is used to mark up structured information in files; information is split into content and roles (Walsh, 1998) in a hierarchy of information. By marking up the script, it is possible for the computer to identify all the content and queries and operations can be performed on this.

The Text Encoding Initiative, an international consortium which “maintains a standard for the representation of texts in digital form” (Text Encoding Initiative, 2014c), have produced a schema for marking up performance texts. This schema is very detailed and effectively renders entire scripts as machine understandable. It includes tags for marking up all the different content types, including but not limited to: scenes, acts, lists of characters, lines and cues (Text Encoding Initiative, 2014b).

This detailed schema provides many different opportunities for further features to be implemented. If correctly marked up, many of the different features within the final project application could be automatically populated by the script. The initial required effort to mark up a script would be considerable. However, the query features would reduce the overall amount of administration for the end user and improve the user experience greatly.
3 Methodology

The project is mainly focused on the development of a mobile application and therefore will follow a standard software development process. The methodologies can be split into three separate areas: design, build and testing stages.

3.1 Design and Build

It has been decided that a modified incremental software development process will be used. Incremental development processes require the overall requirements must first be established. Once these have requirements have been finalised, they are split into smaller objectives for implementation. Each stage is separately designed, developed and then tested before being implemented. Figure 13 (ISTQB Exam Certification, 2013) represents the normal incremental process.

![Incremental Life Cycle Model](image)

The process will be modified slightly by having an extended planning phase. The requirements analysis for the project has been carried out in Section 4. However, before starting development, an overall structure for the application, including a basic prototype UI, will be designed to ensure that all the subsequent features developed are easily incorporated into the application. By taking this extra time to design a system that allows for easy addition of requirements, it should reduce the time required when implementing new features. It will also aid the development of further features as part of further work.

3.2 Testing

When performing final testing over the application, success will be defined through the application’s delivery of the requirements. There will be three different areas considered in testing the delivery success: basic requirements, related work comparison and user testing.

3.2.1 Basic requirements

The application’s features will be compared back to the initial requirements analysis. In section 4 the requirements have been split into mandatory and optional requirements. The mandatory requirements will be checked to ensure that the features provide the described functionality. If any of the optional requirements have been included, then these will also be considered.
3.2.2 Related work
A further area for analysis will be the comparison of the delivered requirements to related works. During the literature review stage a number of related works were identified. These have been critically assessed to understand what further features they could offer. The critical assessment will be applied to the final application to understand if it improves on the related works.

If there is no improvement upon the related works, or the related works offer further functionality, analysis will be carried out to understand why this is the case and whether addressing this could provide opportunity for further work.

3.2.3 User Testing
Perhaps the most important test of success will be the user testing. The application may deliver on the initial requirements, but it must deliver them in an intuitive package.

The testing strategy for this area will be to set a selection of volunteers’ tasks with the application. These will be simple tasks that will require them to navigate the application and utilise the features. If possible, the completion of these tasks will be timed and a questionnaire will be developed to allow for feedback and comments on each of the areas of the application.

User testing has been planned in this manner to try and create a measureable result to understand success. If there are some challenges that users are unable, or are slow, to complete then these will be identified.

3.3 Planning and Risk analysis
Initial planning was carried out prior to the development stage of the project. The Gantt charts from this planning have been included within 0 for more information. The plan was generally adhered to for the whole development process. The main change noted was in the final date for development completion and therefore the date testing was carried out; development was finished a week later than originally intended. However, to ensure this did not impact the final report writing, the user testing was carried out in person and finished by the end of the first week in August.

Risk analysis was carried out before commencing development, which has been included in 0. With the exception of the delay in development, none of the other risks identified occurred. The delay in development was not severe enough to require that the mitigation plan be carried out.
4 Requirements Analysis

The requirements of the application have been split into three categories: system, mandatory and optional.

- **System requirements** are those that will impact on the platform and how the final project application is developed. These are essential to produce a usable application.
- **Mandatory requirements** are the bare minimum of features that should be included within the final application for suitable functionality to be maintained.
- **Optional requirements** will add extra features that would contribute to the overall user experience, but will not render the application unusable if excluded.

The general design of the application is designed to follow an open approach. Whilst there are mandatory and optional requirements, the aim is to build a flexible system platform that will allow for new features to be added with a bare minimum existing code needed edited.

4.1 System Requirements

The system requirements have been considered in detail as part of the Literature review sections on the platform, storage and user interface design. All the system requirements identified, as listed in Table 2, are considered essential for the final application.

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Core engine coded using Appcelerator’s Titanium</td>
<td>● Common core code for iOS and Android</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Only UI code need be written for each platform</td>
</tr>
<tr>
<td>S2</td>
<td>User Interface designed for Apple iPad</td>
<td>● User Interface code designed for Apple iPad</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● All features presented with an intuitive, easy to use UI</td>
</tr>
<tr>
<td>S3</td>
<td>Storage using Couchbase mobile database</td>
<td>● NoSQL database for persistent storage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Flexible schema for future additions to application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Robust back-end support for networking and synchronisation</td>
</tr>
</tbody>
</table>

Table 2 - System Requirements

4.2 Mandatory Requirements

The mandatory requirements, as identified from Section 2.1, have been summarised in Table 3. These requirements are deemed to be the most important to provide the most basic level of functionality for the application.
<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement</th>
<th>Features</th>
<th>Priority</th>
</tr>
</thead>
</table>
| M1  | Script integration, including full mark up support | • XML marked up script  
• Ability to add comments / notes  
• Perform queries to identify specific user lines and items in script | HIGH     |
| M2  | Deliverables status                             | • Present lists of deliverables  
• Ability to provide status updates  
• Ability to automatically populate lists of deliverables from script queries | MEDIUM   |
| M3  | Checklists                                      | • Template checklists for stage manager’s pre and post show tasks  
• Ability to add / remove items from checklist | MEDIUM   |
| M4  | Timer                                           | • Record start time and length of scenes and acts  
• Store show timings | MEDIUM   |
| M5  | Cast and Crew Details                           | • Allow manager to store database of cast and crew details  
• Assign roles to users to group them together | MEDIUM   |
| M6  | Rehearsal Scheduling (email support)            | • Ability to email individuals, from within application, with details of rehearsals  
• Simultaneously email different groups of individuals if required (based on cast and crew details) | LOW      |

Table 3 - Mandatory Requirements

A number of the requirements implementations are based on the success of the script integration; it will be possible to automate the population of a number of the features, such as deliverables, based on queries of the marked up script.

Even if script integration was unable to function as intended, it would still be possible to implement the medium to low priority requirements without the automated aspect.

The main work in providing each of the requirements is the UI design for each of the screens. The actual code will require the manipulation of lists and collections. The features will be implemented in order of their priority.

4.3 Optional Requirements

Due to the time scales of the project, it is not feasible to expect all the initially identified requirements from Section 2.1 be implemented. The requirements in Table 4 are deemed optional for the purposes of the project; their absence will not make the application unusable. As potential future work, these requirements would be implemented in the order of priority in the table.
<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>Scene and Act Charts</td>
<td>• Ability to auto-populate views from list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Schedule rehearsals of scenes automatically through this view</td>
</tr>
<tr>
<td>O2</td>
<td>Networking</td>
<td>• Allow for synchronised data between teams and users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Push changes directly to server, e.g. if updated an item on the deliverables list</td>
</tr>
<tr>
<td>O3</td>
<td>Rehearsal scheduling (calendar view)</td>
<td>• Integrated calendar view within the application to visualise rehearsal schedule</td>
</tr>
<tr>
<td>O4</td>
<td>Blocking design tools</td>
<td>• Built in blocking annotation tools</td>
</tr>
</tbody>
</table>

Table 4 - Optional Requirements

The main reason the charts and calendar views have not been considered mandatory is the extra complexity required in their UI design. Given the opportunity, the chart should be implemented as the first priority as this would be able to interact with the cast and crew details, and script queries, to become automatically populated. This feature being implemented would result in rehearsal scheduling being less manual as groups of users would automatically be grouped by the scene or number to be rehearsed.

Creating network support will result in another layer of coding needing performed, hence why this has been excluded from the mandatory list. However the planned use of Appcelerator’s Titanium as the toolkit should reduce the overall level of work this requires, (see Section 2.2.4) as it includes a system of support for networking.

The blocking design tools would potentially be the most complex extra feature. As noted in the requirements analysis, in Section 2.1, the related work StageWrite already provides this functionality to a very professional degree. If implemented, this feature would need to at least meet the quality of StageWrite or risk being negatively compared. To achieve the professional quality for this feature there would be required a significant time investment. The required time investment is unlikely to be possible given the other features needing implemented in the short time available.
5 Professional, legal, ethical, and social issues

5.1 Professional issues
During development, there is a requirement that the application is developed to a professional standard. The application is intended to be used as a tool within the production of a theatre show; if the application was produced to a poor standard then its use could end up negatively impacting the production.

A script written by TLC Creative was included as an example. By agreeing to include their script, TLC Creative will want to ensure that it has been associated with a professional product. If the application was poor quality, there is a risk it could negatively impact the reputation of TLC Creative.

During testing, the application will be used by various volunteers. This process of testing was carried out in a professional manner, with carefully organised questionnaires and scenarios. This process ensures that time committed by the volunteers was used efficiently.

5.2 Legal Issues
TLC Creative own the copyright to the script “Puss in Boots”. Their permission was needed to use their script in the development of this prototype application.

Having contacted TLC Creative, the permission was granted to use the script providing:

1. A copy of the script was purchased from the publisher
2. A copyright notice is included wherever the script appears stating that it has been supplied for academic purposes and is not distributable to anyone outside the academic community and not available for use without purchase of scripts and rights.

The script was purchased from the publishers by Heriot Watt University and the copyright notice has been included as part of the script view screen. Furthermore, the XML version of the script includes the copyright notice within the header element.

5.3 Ethical Issues
As part of the testing methodology, user testing and questionnaires have been used to gain feedback. The use of this data will be subject to data protection; no identifying personal information will be kept on record. Before carrying out the questionnaire and testing, consent was gained to confirm that the results of the questionnaire could be used for the project. The disclaimer, included in Figure 14 below, was shown to all users before testing was started.

---

Thank you for agreeing to test my application, developed for the purposes of my MSc project in Information Technology (Software Systems).

The application is intended to help in the planning, production and execution of theatre shows. Producing a theatre show is complex due to the level of co-ordination required between the director of the show and all members of the cast and crew. The project application intends to reduce the level of administration burden for the director by providing tools that will automate actions and help organise the different teams in an efficient manner.

The testing will require that you:

1. Carry out a set of defined tasks
2. Fill out a questionnaire regarding the tests done and your thoughts on the usability of the application
3. Carry out a debrief to share thoughts on the application and any of the features
Please note, no personal information will be kept that could be used to identify you. The only information that will be required for the purposes of analysing data is your age.

**PLEASE NOTE: By completing the task and questionnaire you consent to this information being used within my project.**

![Figure 14 - Testing disclaimer shown to participants before testing](image)

Having discussed the ethical issues with the project supervisor, it was not considered necessary to gain written consent from the users given the anonymous nature of the data obtained and the notice included within the disclaimer. The signed Heriot Watt School of Mathematical and Computer Sciences (MACS) ethics approval form has been attached in Appendix 1.

The completed MACS risk assessment form has been attached in Appendix 2. No issues were noted requiring further attention.

A potential ethical issue that could arise surrounds the storage of personal contact information for the contacts feature. This has been mitigated in the initial iteration by ensuring that no data is shared across networks. Furthermore, the data stored as part of the application is sandboxed which ensures that no third party applications can access the data.

However, it should be noted that as part of future work it is recommended that networking will be introduced. In the event that contact data was to be synchronised across the network, action would be taken to ensure that this data was securely transferred. It is also expected that some users would have different levels of access to features within the application. For example, cast members might have no requirement to see all the contact details so this feature could be disabled for these users. Being able to disable certain features for different users will help to ensure that access to the sensitive data is reduced to a minimum.
6 Development Process

The development process was complex due to the scope of the project. As well as the actual application development, the original script was required to be parsed into a valid XML document. The following sections discuss the different areas of the development in greater detail.

6.1 Script Parsing

Before commencing application development it was necessary to parse the theatre script (hereafter “source document”) obtained from TLC Creative Ltd. This was necessary as it provided the foundation upon which the further features would be based.

The source document was obtained in a Microsoft Word 97-2003 Document (“.doc”) format. Per the TEI, for an XML document to be valid, it must be both: valid when compared to a schema and well-formed (Text Encoding Initiative, 2014a).

A computer script was written and developed using Python, to automate the majority of the source document parsing. The computer script used regular expressions on lines of the source document to identify different elements: acts, scenes, songs, spoken lines and stage directions. Having identified the relevant elements, the script would add opening and closing tags and then write these to an output document.

The regular expressions used to identify the different tags were relatively simple. For example, a word before a colon was considered to be the speaking part name and thus words after this were words in the line. Words inside brackets were identified as being stage directions. An example of a line of speech converted from the source document has been included within Table 5.

<table>
<thead>
<tr>
<th>Original Text</th>
<th>Jack: (Sternly) What did I say about cat jokes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marked-up text</td>
<td><code>&lt;sp who=&quot;#jack&quot;&gt; &lt;speaker&gt;Jack&lt;/speaker&gt; &lt;l&gt;&lt;stage&gt;Sternly&lt;/stage&gt; What did I say about cat jokes?&lt;/l&gt; &lt;/sp&gt;</code></td>
</tr>
</tbody>
</table>

Table 5 - Example conversion from plain a text script line to XML marked-up version

The regular expressions were performed at a line level, and therefore some elements which spanned many lines needed to be held in memory. For example: acts, scenes and songs. If one of the multiline elements was noted, then a Boolean value for the element was set to true. Any of the specific Boolean variables being true would ensure a check was performed to make sure that closing tags were applied before opening a new one.

For the final document to meet the specification of the TEI schema it had to include a header element. It is within this header that elements that describe the properties of the source document such as the: title, authors, cast, synopsis, etc. It was not possible to parse these elements from the source document automatically as there is no regular expression that can be used to identify them. Therefore, these items were added to a separate file manually before being concatenated to the output from the computer script.

One of the main issues noted with this process, beyond the manual work involved, was the formatting of the source document; the source document had to match the defined regular expressions exactly, otherwise the parse would fail. Errors in formatting of the source document, such as a colon being excluded at the beginning of a line, meant that elements were not correctly converted as expected.
If the application were to be developed further and released to the general public, there would need to be some development around the parsing of scripts due to the complexities involved. This would either need to be provided as a service, or there could be further development to try and create tools that users could utilise to convert scripts themselves.

One issue that was not resolved during development was the specific mark-up for props. The TEI schema currently doesn’t include any definitions for props; the schema would need to extended to include these definitions. The only impact this had on the application was that it wasn’t possible to query the script for its props to automatically populate checklists.

6.2 Titanium Framework and Development Process

Development of the application was performed using the Appcelerator’s Titanium software development kit and framework. The primary reason for using this framework was to allow for easy conversion of the application into Android, whilst only needing to update the UI.

In general, the framework was found to be a powerful tool for creating a native application. In retrospect, there were a few issues found in the implementation that would need to be addressed for the framework to become a true competitor for wholly-native built applications. The main issue identified is that the framework very quickly becomes out of date. The Appcelerator development team have to base their work around the available technology from Apple and Google. This creates a lag between releases from Apple and Google and implementation within Titanium. New iterations of the Apple and Google frameworks were announced shortly after commencing work on this dissertation (Apple Inc., 2014c, Google Inc., 2014c). If the author had been developing the application using the native development kits, then new features could have been implemented rather than being restricted.

The other issue noted was the steep learning curve to be able to use the SDK. There is very little in the way of documentation available beyond the API documentation and user question and answer forums. This made the initial development attempts inefficient as different approaches had to be attempted until the best solution was found.

One of the main strengths noted was that Titanium applications are designed around a Model-View-Controller (MVC) framework. Each view that can be seen in the application has its own UI, style and controller code. It is also possible to link data models to specific views, this allows data driven updates, without explicitly needing to code this logic in the controller class.

The use of Alloy for populating data models is one of the most useful tools available within the Titanium framework and has been used extensively throughout the application. Alloy uses Backbone.js to automatically populate views with information passed in from models, which in this application is an SQL table. Backbone.js is a technology that allows for data models to be tied to specific views via “key-value binding” with automatic event handling for the model being updated (DocumentCloud, 2012). In practice, this allows for one view to be created for one item in a model; the view will then be repeated for each of the items in the model. There are also useful functions for querying and manipulating the items or collections of items in the model. These functions can filter out results that are populated in views or can even be used to manually populate views.
The overall design of the application has been presented in Figure 15. The MVC framework was found to be very powerful for designing the flow of the application as features were easily separated into their own MVCs.

![Figure 15 - Overview of the Panto Genie design](image)

The features all share access to utility functions, which are used to perform actions such as query the script, and all features interact with the database. Each of the features is accessed in-application by the main tab controller, as per Figure 16.

![Figure 16 - Panto Genie screenshot - Example of the Tab bar for accessing features](image)

### 6.3 Script view

The first and most complex feature that was developed was the script view. This view traverses the XML script document and then presents it on the screen. The script view also includes tools for navigation, part highlighting and comments. The complexities in the development arose around the presentation of script on screen.

The script rendering consists of three TableView elements which are presented separately using ScrollableViews. A ScrollableView is a UI element for presenting a collection of views where the user can swipe through the different views horizontally (Appcelerator, 2014g). A TableView is a UI element for presenting tabular data organised within rows, this is traversed using vertical scrolling (Appcelerator, 2014h). The TableView was used for presenting the script on screen, where each line of script is its own form of data and can naturally be presented as a row. The other benefit of using the TableView to organise the layout meant that complex operations regarding the height and placement of the different UI elements was not necessary.

The script is rendered by traversing the elements in order, defining which type of element they were (based on the tag and attribute values) and then creating a table row and adding it to the relevant TableView. Functions were developed to allow traversal of the XML file from a given a location in the
script going either direction in the XML hierarchy. Reverse traversal is required for when the user swipes from left to right and previous pages are then loaded. All traversal is done from an index location in the XML and for the previous page this starting index is the starting point on the next page.

Due to the size of the XML script it was not possible to parse and present the entire file for viewing. It would take a significant amount of time to parse the whole document and then render the elements on screen. The approach taken to solve this issue was to parse and render three pages of script at a time. This traversal loop would carry on till the final element of the script was located. Therefore a break is called when the height of the elements on the page would be greater than the size of the page. Figure 17 below visually represents how pages are parsed from the script; with the direction of script traversal and the index point from which rendering is performed.

One of the major drawbacks of using the TableView to present the script pages is the inability to know the dimensions of the added rows until after they have been laid out in the table. After being rendered, a PostLayout event is fired by the new row which must be caught by a listener. Only at this point is the dimension information available. The difficulty is that it is not possible to know when the PostLayout event will be fire and there is no way to force it to be fire. Consequently there is no reliable method to ascertain if rows added will cause the table to be larger than the viewport, and thus be presented off screen. It is quite likely that the PostLayout events will not be fired until after rows have been rendered off screen. Normally rows being rendered off screen would be adequate as vertical scrolling is used to navigate the view. However interacting through vertical and horizontal scrolling feels unnatural and does not feel like a metaphor for a real script. This implementation would not follow iOS design guidelines for metaphors (Apple Inc., 2014a).

As an initial work around for the issue of rendering script off screen, a simple fix has been applied to the traversing layout loop. Before adding any items to the TableView, an image is created of the new row of which the height can be checked. Each of the image heights is cumulatively added and if the new row were to take the cumulative total above the height of the page then the row is not added and the traversal loop is exited. The author does not feel this is an optimal fix for two reasons: there is not a 1:1 correlation between the height of the rendered image and the actual rendered row, this requires crude adjustments be made to ensure a rough match; and secondly, it is relatively costly operation to generate an image for every row rendered. Despite the fix not being perfect, the script is usable and responsive despite the operations that occur in the background.
6.3.1 Part Highlights and Contents

Other features available within the script module are: the ability to highlight the lines of specific roles when being rendered on screen; and, the option navigate directly to specific acts, scenes or songs in the script.

The role and contents items are parsed directly from the script using utility functions in a CommonJS file. CommonJS is the recommended approach for sharing functions across different controller files (Appcelerator, 2014c). CommonJS is functionally equivalent to static classes in languages such as Java. CommonJS has been used for the utilities functions as there are many different features where they are necessary. This approach reduces code duplication and will help promote future development by ensuring more concise, maintainable code.

The contents items are parsed manually by traversing the script for div elements. These elements are then identified as being contents items by querying the type attribute. Once identified, the contents elements are stored in a JSON object with: the type of contents item (e.g. act, scene or song), the index position within the XML document and the text of the title. This information is then presented within a drop-down on the script view, as per Figure 18.

![Figure 18 - Panto Genie screenshot - Example of the contents picker for the script view to allow easy navigation](image)

Despite the processing cost of traversing the entire script for contents items, the query must be performed in this manner as the index of the relevant elements must be obtained. By providing the index element, the script can be rendered from the chosen contents item. To reduce the number of times this costly operation is performed, the JSON object is stored within the application memory. Given the contents of a script will not change, unless further scripts are added, it is not necessary to perform this operation more than one time for a given script.

Cast details are held within a div element, with type attribute “Dramatis_Personae”. This element is located using built-in XML functions. Unlike other queries, no index position needs returned and therefore manual traversal is not necessary. When the “Dramatis_Personae” element is returned, the roles elements are obtained and stored in memory. These cast items are stored as a JSON object.

When a character is chosen to be highlighted, the character ID and chosen colour are stored within the local application database. As part of the script rendering process, the ID of the currently line speaker is queried against the database and if it is found then the line’s background colour is set to the colour chosen by the user. The lines are then rendered as per Figure 19.
6.3.2 Comments

The comments function was added to allow the user to mimic writing notes in the margin of a script. This is inspired by the iOS design philosophy for metaphors (Apple Inc., 2014a). Comments can be added to the script via LongPress event listeners attached to the rendered lines. A LongPress event occurs if the user presses on the screen for longer than a single click. After LongPressing, an input box appears to allow for the user to enter a comment related to the line.

The option to use templates for typical comments is also available; such as stage directions and positions. There is a separate utility function that has been used to return the comment templates if the user requests the list.

Comments are rendered on screen in the right hand margin of the script, as per Figure 19, and where there is more than one, they will be concatenated together with a comma and space delimiter. If comments are too big for the margin they will be ended with an ellipse and the user can then click to see the full comments.

Comments and the index position of their line are stored in a SQL database. When rendering the script lines the index within the XML source is appended as an attribute to the label; this index position is passed as part of the LongPress event. When rendering the information on screen, the comment database is queried for any comments with the same index; if any results are returned by the query then these are rendered on screen.

6.4 Checklists

Fundamentally, the checklist feature simply provides a UI to database interaction. Two databases used to store the checklist data: one database stores the overall checklist group title as a primary key; and checklist items are stored in a separate database, where a foreign key is used to link items together under the same checklist group. Alloy modelling, as discussed in section 6.2, is used to present the data from the checklist group database when first navigating to the checklist tab. If a user clicks on this item, they are taken to a detailed view which lists all of the items belonging to the group.

When a detailed view is selected, the name of the group is passed as an argument to the detailed view controller. The group name is queried against the detailed checklist database to return a collection of items in the same group. Checklist detail items belong to a sub-section in addition to the overall checklist group. The sub-sections are used to group detailed items together within different sections. Alloy modelling currently doesn’t support auto-population within separate sections of a table; therefore the detailed items view is populated manually. Example checklist items have been included in Figure 20 where the tick is rendered as either transparent or solid depending on the status of the item.
The manual population of the checklist detail view is performed by looping through the items returned by the initial checklist group query. Each item populates a template row and then grouped added to an array of arrays, based on the subsection. The array is passed to the main TableView which automatically renders the rows on screen.

6.4.1 Adding New Items
The process of adding new items is common across all the features. However, to aid in explanation it will be explained in the context of adding a new detailed checklist item.

To add a new detail item, the user must set the view to edit mode by using the top right navigation bar button. Edit mode allows the user to delete or add new items to the model. To add a new item, the user clicks on the plus symbol located where the edit button was. This opens a modal view where the user can fill out details and then save the item.

The add view appears as a form, as per Figure 21; the form is a formatted TableView, with custom template rows. The template rows are designed as separate controllers, which are then imported into the current view using Common.js. This allows each of the different features to utilise the same rows easily and keeps the design consistent. Each row consists of a label and then either a: text input, a text area input or a custom drop-down input.

Drop-downs pickers are not available within the Titanium framework; therefore a custom version has been developed. The custom drop-down consists of a text input with a downward pointing arrow button rendered at the right hand side, see Figure 21. The text input has a touch event listener attached to the image; if the button is clicked then a popover appears with the appropriate selector for the input row is displayed. The input can still be used as a standard text input if applicable. For the checklist detail, a drop-down is used to present the possible sections that a new item can be assigned to.

6.5 Contacts
Contacts related to the performance can be stored within the application using SQL. Stored contacts are displayed automatically in the contacts view using Alloy and Backbone.js. The rows are populated with the contact’s name and their role within the performance, as per Figure 22.
Currently there is a placeholder for a picture of the contact; with further work the images of the contacts should be stored and presented.

![Contacts](image)

Figure 22 - Panto Genie screenshot - Example of the contacts view with placeholder contacts and search bar

If a contact is selected, a detailed view with the contact information is presented. When creating the detailed contact view, if a contact ID is passed as an argument, the database is queried for this contact and any information returned is automatically populated.

The contact information is added and shown using template rows as detailed in Section 0. Many of the rows use the standard text input, where the only variation has been the type of keyboard input used. The keyboard type changes depending on the information being requested. For example: text inputs, such as names, require a standard keyboard; number inputs, such as telephone numbers, required a numerical input. The different keyboard inputs have been specified to ensure the most convenience for the user.

The contact’s role information can either be input using the standard text or the drop-down picker. The drop-down picker is populated with a number of standard theatre performance roles, as identified in section 2.1.1, but also includes the names of characters identified from the script. These characters are populated using the same utilities function as used in the script highlight feature discussed in Section 6.3.1. An example of the picker with populated details can be seen in Figure 23. By using the template picker the character’s unique ID is stored with the contact. By linking a contact and a character as such, the rehearsal feature can use this knowledge when organising rehearsals. This is discussed further in Section 6.6.

![Add New Contact](image)

Figure 23 - Panto Genie screenshot - Example of add new contacts screen with automatically populated drop-down picker for contacts
A convenience feature added to the contacts view was the contact search function, which can be seen at the top of Figure 22. Search panels are added to TableViews by specification in the Alloy XML mark-up. The default search value is the title of the row. However, it is possible to specify a custom search attribute and this has been set as a concatenation of the user name and role as these are the items that appear on the contact’s view.

6.6 Rehearsals
The rehearsal feature allows the user to organise, view and share rehearsals. The rehearsal feature utilises a number of different technologies to deliver full functionality: rehearsals are saved within the application using the SQL database, they are also saved to the calendar within the iPad and they are sent by email in an iCalendar format to specified contacts.

Creating a rehearsal used the same standardised input rows as used in the other features and the user is asked to specify: a rehearsal title, start and end time, the scene being rehearsed and contacts.

The start and end time inputs utilise the custom popover input, where the native Apple date picker is shown to help select the time, as per Figure 24. The native date picker helps increase consistency with other applications. Consistency is considered a key design philosophy by Apple, as discussed in section 2.3.1.

![Figure 24 - Panto Genie screenshot - Example of native date picker being used with the custom drop-down picker](image)

The unique functionality offered by this feature is the script integration. The user selects the scene to be rehearsed using a drop-down picker. The drop-down is populated with all the acts, scenes and songs which are populated using the same content parsing utility discussed in section 6.3.1. The content parsing utility returns section’s title and index. The index is passed to another utility function which traverses any element at an index, in this case one of the contents elements, and returns a list of any included character IDs. The set of IDs is queried against the contacts database and if contacts are found with a matching role, then their details are automatically populated in the rehearsal contacts’ list and added to a collection of invited contacts. If contacts included in the contents section are not found, then an alert is popped up to warn the user that these parts do not have a matching contact stored within the application.

Manual editing of contacts can also be performed and is done through the use of a TableView showing all the invited contacts as per Figure 25. Users are able to delete contacts from this, but if the user clicks the button to add contacts, the table view is cleared and repopulated with a list of all
the contacts who are not already invited. If the user adds any new contacts, these are added to the collection of invited contacts and the on-screen list is updated.

6.6.1 Saving Rehearsals

If the user clicks save, then the rehearsal is automatically saved to the local iPad calendar. When the user first opens the application permissions to access the calendar are requested. The user must also select which calendar is used at the point of providing permission. Requesting permission is required due to Apple security, but allowing the user to choose the calendar, allows the user to feel in control which is one of the key Apple design guidelines (Apple Inc., 2014a).

Saving events to the calendar uses an Appcelerator API. Once the event has been saved in the calendar, the ID created by the built-in calendar is used as the primary key within the application’s rehearsal database. The linking of IDs allows the event items to be kept in sync between the application and the built-in calendar. The start and end dates of the calendar are not stored within the application; this was a design choice to mitigate the possibility that the user were to update the entry using the built-in calendar. Before presenting rehearsal data, these times are queried from the calendar.

To enable the sharing of the rehearsal to invited contacts, upon clicking save an email dialogue is opened. The dialogue is opened with all the relevant fields pre-populated. The rehearsal details are saved as an .ics file and attached to the email. A .ics file is used to present iCalendar data, as specified by The Internet Engineering Task Force (2009), which is the standard format for sharing event information. As this is a standard format, it ensures the details can be parsed by many different calendar programmes. There are no API functions available for creating .ics files in the Titanium framework, therefore a custom function was developed to manually create this file according to the standard (The Internet Engineering Task Force, 2009).

When rehearsals have been saved, they are displayed in the main rehearsal view using a TableView. The Alloy modelling has not been used to display the rehearsal events automatically as the events are grouped into different sections by time categories. The different categories and overall presentation can be seen in Figure 26. Manual sorting was required as Alloy modelling does not work with Sections in tables. Furthermore, events must be caught if the user directly edits rehearsals in the built-in calendar view. If the built-in calendar events are caught, it triggers a manual refresh of
the TableView; the events do not pass detailed information, therefore the whole collection must be refreshed (Appcelerator, 2014f).

The main complexity with presentation of the rehearsal events was creating a function to ensure: the rehearsal entries are sorted chronologically by start date, grouped in the correct section and that rehearsals in the past are not shown. The correct section was ensured by comparing the start date of the event to a series of dates; these dates correspond to the dates of the sections and are refreshed daily. Once the correct section has been found, a loop through section items is performed to find the index where the start date of the section item is before the item being checked; the item is inserted at this index point. To ensure the dates are not in the past, the first check is that the rehearsal entry does not come before the 00:00 hours on the date of comparison.

![Upcoming Rehearsals](image)

**Figure 26 - Panto Genie screenshot - Example of a rehearsal item in the main rehearsal view, categorised by time**

### 6.7 Timer

Timers are stored in groups and therefore the timer feature uses a similar database format as the checklist; one database is used to store the group titles and then a further database is used to store the timer details.

When creating a timer group, it is possible to auto-populate the timers based on the scenes in the script. The auto-population is performed by leveraging the information returned by the content parsing utility, as discussed in section 6.3.1. For every scene in the contents collection, a new timer is created with the scene title and added to the database model. The new timer is automatically populated with the group title to ensure it is included within the correct timer group.

The Appcelerator API does not provide any APIs functions for timing; this function has therefore been designed manually. The function operates by calculating the time elapsed as the difference between the Coordinated Universal Time (UTC) milliseconds at the date of calculation and UTC milliseconds at the moment the timer was started; an interval is set to before calling back this function and updating the time elapsed label. This approach is based on a solution found in the Appcelerator Community Questions & Answers forum (Baetica, 2014). The solution has been adapted to respond to button clicks and provide more information to the user on-screen.

The start time and status of the application – either: new, started or stopped – are stored as part of the timer database. The statuses are recorded to allow the user to navigate away from the timer page whilst still having the timer functional. If the user returns to the timer page with an active timer, upon loading the timer collection it will set up the timer to carry on counting as if the user had never left the page.

Individual timers are represented as rows which include the functionality to start, stop or reset the timer as necessary. Examples of the three types of timer are shown in Figure 27. As the status of the timer changes, the buttons for start, stop or reset are expanded and hidden as necessary. This design was thought to help keep the UI simple and easier to use. Each of the rows have the same controller code attached to the row, this allows the timers to work independently of each other.
<table>
<thead>
<tr>
<th>Timer Groups</th>
<th>Scenes</th>
<th>Edit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prologue</strong></td>
<td>Started: 4:33:44 PM GMT+1</td>
<td>00:00:12</td>
</tr>
<tr>
<td></td>
<td>Ended:</td>
<td></td>
</tr>
<tr>
<td><strong>Act One: Scene One: The Village Square</strong></td>
<td>Started: 4:33:46 PM GMT+1</td>
<td>00:00:08</td>
</tr>
<tr>
<td></td>
<td>Ended: 4:33:54 PM GMT+1</td>
<td></td>
</tr>
<tr>
<td><strong>Act One: Scene Two: The Royal Palace</strong></td>
<td>Started:</td>
<td>00:00:00</td>
</tr>
<tr>
<td></td>
<td>Ended:</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 27 - Panto Genie screenshot - Example of the timers in the three states*
7 Evaluation

Following development, evaluation is required to understand the level of success of the project. Based on the project guidelines, success within this context can be defined as meeting the following criteria (Heriot Watt University, 2012: pg. 4):

1. Application has realised the project objectives
2. Tested and evaluated as meeting the customer needs
3. Providing a reliable and functional solution

The project objective was to produce an application to help in the planning, production and execution of theatre shows. The author feels that the application produced realises this objective. The actual application provides functionality to help in all stages of a theatre show. Further evaluation was carried out to evidence these claims and also to ensure the second and third criteria have been met.

Further evaluation consists of: a comparison of delivered requirements to the original requirements, usability testing and comparison of the application to related works.

The requirements comparison provides insight into project’s success at meeting the first point and second criterion: the requirements were considered necessary for achieving the project’s objective and the second criterion is met implicitly given the requirements were based on an evaluation of the potential customer needs.

Usability testing allows a sample of users to test the application for reliability and functionality, thus meeting the third criterion.

Comparison to related works further evidences the success of the project meeting the second criterion; it provides a minimum baseline for existing implementations which the application must meet.

7.1 Requirements Comparison

There were three types of requirements specified during the research phase of the project: system, mandatory and optional. The research phase was carried out prior to the development phase and the initial requirements were borne out of analysis of user needs. If these requirements have been met, this provides a simple proxy for understanding how successful the application at meeting the project objective and the success of delivering the customer needs.

It should be noted, that due to the time constraints, the optional requirements have not been implemented within the final project. The optional requirements had already been deemed non-essential for fulfilling the user needs and accordingly their absence has not been considered a negative indicator of the project’s success. However, in the face of the development and feedback from users, the optional requirements have been reevaluated as part of the discussion into future work in Section 8.

Table 6 details the status of the requirements where there ID links the items back to the original requirements analysis. Where these have not been met, a brief discussion follows to explain why this is the case. As part of this discussion, the impact on the users’ needs has been analysed to understand the impact of the project’s success.

<table>
<thead>
<tr>
<th>ID</th>
<th>Identified requirement</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Core engine coded using Appcelerator’s Titanium</td>
<td>Yes</td>
</tr>
<tr>
<td>S2</td>
<td>User Interface designed for Apple iPad</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The majority of the requirements have been fully delivered which further helps evidence the realisation of the project objectives. However, it should be noted there are two areas that have not been delivered at the point of this report; this is the result of a conscious change in plan and, as per the discussion below, is not thought to represent a failure to meet customer needs.

### 7.1.1 Couchbase Storage

The planned use of Couchbase for the storage system was initially considered due to the fact it allowed for the use of NoSQL technology and thus the: flexible schema when providing future updates, robust networking functionality, and ease of integration with the Appcelerator’s Titanium platform due to developer published plug-ins. However once development started these initial considerations were reconsidered and it became apparent that, in practice, the existing Alloy support for SQL would better meet the customer needs.

The main issue noted was the level of Titanium platform support available. During development it was found that Couchbase’s Titanium support extended to a beta plug-in, which at the date of writing this report had not been updated for eight months (couchbaselabs, 2014). It was found that, from examination of the source code, that the Titanium Alloy framework was not supported in the beta either (couchbaselabs, 2014). As discussed within the development section, Section 6.2, Alloy provides support to render SQL databases quickly and reliably. This was considered an invaluable development tool and on this basis, Couchbase was not developed any further.

The loss of the other NoSQL advantages are mitigated by the Titanium framework. Being unable to use flexible schemas is offset by being able to perform database migrations within the Alloy framework; migrations allow for easy SQL schema updates by incrementally changing existing entries to match the correct schema (Appcelerator, 2014a). The use of the Alloy framework and SQL means that the application can no longer use the Couchbase networking services; however, it is possible to utilise Appcelerator’s own robust networking services without reducing the user functionality.

Providing further justification for the switch to SQL, Couchbase support for Titanium was no longer supported at the date of this report. Only Xamarin and PhoneGap are advertised on the Couchbase site as being the cross-platform development technologies supported (Couchbase, 2014a).

The customer needs will not be negatively impacted by the decision to use SQL over NoSQL. Alternative approaches have been identified which fulfil the initial requirements identified in the research phase. Furthermore, given Couchbase is no longer supported on the Titanium platform, the user should actually be positively impacted; they will not be interfacing with an unsupported beta product and will benefit from the fast and reliable Alloy implementations for presenting SQL data.

### 7.1.2 Deliverables Status

It was originally intended that there be support for deliverables within the play. This was planned to be an integrated method for sharing the status of the deliverables between members of the crew and the director. However, once development started, it was noted that the checklist feature would provide the same basic level of functionality; a list of items that need to be checked off and...
completed. On this basis, it was concluded that the two features should be combined into just the checklist feature, where eventually checklists could be synchronised between devices.

While the two features have been combined, it should be noted that the intended functionality of the deliverables requirement has not been provided. There is no option to synchronise the checklist across multiple devices and no option to provide updates on checklist items. The synchronisation feature has been excluded as it wouldn’t be possible to implement in the application whilst there is no underlying network support. Consequently the item updates feature was excluded to manage the user’s expectations. Had this been implemented, the user might have expected that they would be able to share these updates with other users. By explicitly reducing the functionality the expectations of the user will be lowered and this will help to drive user satisfaction.

The ability to explicitly provide updates on checklist items could be easily implemented as notes within the checklist. If items were clicked on this could open up a detailed view where there could be a free text box for writing updates. Once networking is supported, as the items are updated and synchronised, this would automatically be shared.

Given the similarity of the two checklist and deliverables features, the grouping of the two should not greatly impact the users’ needs; it is still possible to create lists of deliverables items. However, being unable to share and provide updates does present a reduction in what was originally intended. It is expected that if attempts had been made to develop these functions fully, the end result would have been flawed without the underlying networking support. Excluding the extra features was deemed the reasonable approach to ensure the application was reliable and functional as a solution. These features should be revisited as part of future work.

7.2 Usability Testing
Usability testing has been carried out on a sample of 10 users. The age and skill of the participants was varied and accordingly, there were some variations in the perceived usability of the application. In general, the application was very well received with an average System Usability Score (SUS) score of 79. The SUS score of 79 is above the average score of 70 (Bangor et al., 2009), and as can be seen from the grading system included in Figure 28, is considered: a very high C grade, is six points away from being considered “Excellent” (Bangor et al., 2009).

![Figure 28 - The SUS scoring system and grading (Bangor et al., 2009)](image)

SUS is an “industry standard” tool for measuring how usable a system is (usability.gov, 2013). The main difficulty arises that that due to its subjective nature, measuring the usability of an application is not an exact science. However, SUS has been cited over 1300 times since it was created by John Brooke in 1986 (usability.gov, 2013), which provides strong evidence supporting its worth as a tool for usability measurement. SUS is measured through the results of a questionnaire with 10 standard questions. Responses are given as one of five standard answers that range from “Strongly Disagree”
to “Strongly Agree” (usability.gov, 2013). The SUS questions, based on Brooke’s original publication (Brooke, 1996), have been included in Table 7.

<table>
<thead>
<tr>
<th>No.</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>I think that I would like to use this system frequently.</td>
</tr>
<tr>
<td>2.</td>
<td>I found the system unnecessarily complex.</td>
</tr>
<tr>
<td>3.</td>
<td>I thought the system was easy to use.</td>
</tr>
<tr>
<td>4.</td>
<td>I think that I would need the support of a technical person to be able to use this system.</td>
</tr>
<tr>
<td>5.</td>
<td>I found the various functions in this system were well integrated.</td>
</tr>
<tr>
<td>6.</td>
<td>I thought there was too much inconsistency in this system.</td>
</tr>
<tr>
<td>7.</td>
<td>I would imagine that most people would learn to use this system very quickly.</td>
</tr>
<tr>
<td>8.</td>
<td>I found the system very cumbersome to use.</td>
</tr>
<tr>
<td>9.</td>
<td>I felt very confident using the system.</td>
</tr>
<tr>
<td>10.</td>
<td>I needed to learn a lot of things before I could get going with this system.</td>
</tr>
</tbody>
</table>

Table 7 - Standard SUS questions (Brooke, 1996)

Before a user can complete the SUS questionnaire, they are required to spend time with the application so they can form valid opinions (Brooke, 1996). To allow the users to form an opinion of the application, they were given a series of representative tasks to complete. These tasks were given in the form of user story points and covered all the features of the application. Appendix 5 includes details of the tasks carried out by users, including details of how they link back to the application and features being tested.

A debrief was performed with the participants after completing the questionnaire; this is not part of SUS but was included to get some detailed responses related directly to the application. The debrief responses helped to identify any missing features, rather than just assessing the usability of the application.

7.2.1 Analysis of Results

As noted, there were 10 participants in the user testing. Given research has found that between 8 and 12 users can provide reliable results (Tullis and Stetson, 2004) a sample size of 10 was deemed sufficient for analysis. The age of the participants varied from 24 to 64 and included both males and females. The spread of participants provided a mixture of proficiency in computing skills; the spread of skills was useful for seeing where the application was familiar to regular users and adversely, where extra information would be necessary for the less proficient users. A portion of the testers were chosen for their experience in the theatre industry, to provide deeper insights into the success of the features during the debrief. The summarised results have been included in Table 8 below, for the full detailed results see Appendix 6.
<table>
<thead>
<tr>
<th>User</th>
<th>Age</th>
<th>Gender</th>
<th>Total time</th>
<th>SUS Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>M</td>
<td>15:57:00</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>58</td>
<td>F</td>
<td>24:46:00</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>58</td>
<td>M</td>
<td>19:32:00</td>
<td>85</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>F</td>
<td>10:48:00</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>M</td>
<td>8:40:00</td>
<td>93</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>M</td>
<td>10:52:00</td>
<td>85</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>M</td>
<td>10:43:00</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>61</td>
<td>M</td>
<td>12:55:00</td>
<td>85</td>
</tr>
<tr>
<td>9</td>
<td>63</td>
<td>F</td>
<td>27:58:00</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>27</td>
<td>M</td>
<td>9:46:00</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Average</strong></td>
<td><strong>15:11:42</strong></td>
</tr>
</tbody>
</table>

Table 8 - Summarised Usability Testing Results

To help aid the readability and comparability, the SUS scores have been plotted on a bar chart, see Figure 29 below. From analysing the results, there are two users, User 2 and User 9, who found the application significantly less usable than the other participants, scoring 27% and 37% below the average respectively. Further analysis has been undertaken to understand why this was the case. It is expected that if the reasons for the lowest scoring users can be understood, then this is where the most value can be derived in terms of assessing flaws in usability.

![Figure 29 - Bar Chart presenting SUS Score by User](image)

Both the low scoring participants were females who had an average age of 61; during debrief discussions, both of these participants stated that they were not technically proficient or experienced using iPad applications. None of the other participants had similarly low experience levels; this would suggest technical proficiency was the driving factor behind the perceived usability. From previous studies performed regarding SUS, the gender of the participant has no impact on the perceived usability (Bangor et al., 2009) and therefore hasn’t been considered within the analysis.

Studies performed by the British Government’s Department for Business Innovation and Skill (2012) found that there is a link between age and frequency of computer usage for skill levels in Information and Communication Technology (ICT). Most relevant in the findings was the conclusion...
that those aged between 55 and 65 had below average ICT skills. Based on these results, it is perhaps not surprising that the deemed usability of the application was lower for User 2 and User 9. Despite being in the same age bracket, the comparative frequencies of computer usage for User 3 and User 8 would appear the key factor in differentiating them from the low scoring users.

It could be argued that these users should be excluded from analysis, these users are unlikely to use an iPad and therefore it is unlikely an application such as this would present an attractive prospect for them. However, the author has deemed it useful to include the users within the analysis as it highlights the least usable aspects of the application. If the application was made more usable to the least technically proficient users, then the target audience would have a higher perceived usability overall.

7.2.2 Conclusion on SUS results
Despite the lower perceived usability of User 2 and User 9, it is important to note that the participants were still able to complete the tasks assigned with some limited help. It was noted that the initial interactions with features and processes were what caused the most difficulty; rather than finding the overall process too difficult to complete. For example, multiple tasks required that the user deleted certain items from collections. Initially the process of deleting an item required some guidance; however, when subsequently approaching similar tasks, the participants were more confident and quicker at completing them without guidance.

Based on this observation that users quickly adjusted to the application after limited experience, there are two key findings:

1. The application follows the iOS design guideline for consistency which is proven to aid usability (Apple Inc., 2014a)
2. Perceived usability would increase in the face of built-in help or tutorials, to help bridge the initial knowledge gaps

The first finding is important as it provides confirmation that the design approach has followed the iOS design guidelines published by Apple (2014a). Ensuring the design followed these guidelines was one of the system requirements identified in the research phase of the project and further helps evidence the requirements were delivered.

The second finding is considered key for the focus of any further work performed. The absence of a tutorial was one of the key pieces of feedback from all users as part of the testing debriefs and would therefore be a useful and popular feature. It is the author's belief that given the opportunity to follow a tutorial when first using the application, and then being able to access a help function at later points, users would experience would have far fewer usability issues.

As part of the debrief users were invited to highlight any difficulties they face in usability and provide insight into how successful they felt the features were. The responses were positive, however a number of smaller items were noted which would require attention before the application could be released. All the main items identified from feedback have been included in Table 9.

<table>
<thead>
<tr>
<th>Bug/Improvement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improvement</td>
<td>Pop-overs are sensitive to being closed if user accidently clicks outside the pop-over. Perhaps would be better placed as a modal window</td>
</tr>
<tr>
<td>Improvement</td>
<td>A smoother experience would be provided if the application showed new highlights automatically after creating them</td>
</tr>
<tr>
<td>Improvement</td>
<td>Loading for contents drop-down is slow (provide feedback for loading as implemented for the content’s pop-over in the script view)</td>
</tr>
</tbody>
</table>
## Improvement
Change rehearsal end time automatically if changing the start time

## Improvement
When commenting, provide a visual cue for the line currently being highlighted

## Bug
Hint text obscured in drop-down input field (would be solved by instructions in a modal window)

## Improvement
Import contact details from iPad contacts (reduce manual typing if contacts are pre-existing within the iPad)

| Table 9 - Description of feature improvements and bugs noted from users' testing debriefs |
|--------------------------------|----------------------------------------------------------------------------------|
| **Improvement** | **Change rehearsal end time automatically if changing the start time** |
| **Improvement** | **When commenting, provide a visual cue for the line currently being highlighted** |
| **Bug** | **Hint text obscured in drop-down input field (would be solved by instructions in a modal window)** |
| **Improvement** | **Import contact details from iPad contacts (reduce manual typing if contacts are pre-existing within the iPad)** |

Each of the items identified as improvements would require relatively small changes in the code to fix. Given the nature of the difficulties faced by the participants – they were changes to improve the experience rather than application breaking bugs – this should be taken as highly positive feedback for a first round of user testing.

In general, given the excellent results of the usability testing and the nature of any feedback, the application should be considered both a reliable and functional solution. This indicates that the application has successfully met the second criteria noted for being considered a success.

## 7.3 Comparison to Related Works
There are two related works identified within the requirements analysis in Section 2.1 that have similar features to those implemented in the project application: Rehearsal 2 and ShowTool SM.

Rehearsal 2 provides the ability to: view scripts, highlight lines and add comments. Each feature is also provided in the project application, but with distinct advantages when compared with Rehearsal 2. The two key advantages are:

1. Highlighting is performed for the whole script automatically
2. Comments can be viewed whilst simultaneously viewing the script.

The first advantage reduces the number of repetitive tasks that users are required to perform to achieve the same end result. By automating menial tasks the user is quickly able to gain value from their time by focusing on the aspects that provide value.

The second advantage allows the user to simultaneously view the script and comments at the same time. This eases user interaction but is also a closer metaphor for a real script. Effective use of metaphors was one of the iOS guidelines that was considered necessary to meet for the project requirements (Apple Inc., 2014a).

A possible disadvantage is the format of the underlying script; Rehearsal 2 is able to import any script as a PDF document whereas the project application requires that the script be in the XML format. This could prove a disadvantage if there found to be significant difficulty in converting scripts. However, the advantages that exist due to the use of the XML format mean that it would be very desirable to solve any issues in this area.

ShowTool SM provides tools to create checklists and record show timings which are also provided within the project application. When comparing ShowTool SM to the project application, the main advantages of the project application are that:

1. Show timers can be automatically populated for timer groups
2. Templates exist for the checklists.

As with Rehearsal 2, these advantages help reduce the time spent on low-value menial tasks.
As well as individually providing more advantageous features than the related works, the project application itself provides one solution rather than requiring the use many different toolsets. This integration of features within the one application allows for the different features to leverage off each other; for example, the timer tool is able to utilise the content parsing utilities created for the script view. This overall helps the application to better meet the objective of the project to provide an effective tool for each stage of the theatre production process.
8 Further Work

As part of the research phase optional features were identified to be suggested as further work. However, following development and testing phases, there were further areas noted that would also add further value to the user.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fix bugs identified from testing</td>
</tr>
<tr>
<td>2</td>
<td>Contact Import feature</td>
</tr>
<tr>
<td>3</td>
<td>Tutorial / Help system</td>
</tr>
<tr>
<td>4</td>
<td>Implement options for changing scripts</td>
</tr>
<tr>
<td>5</td>
<td>Rehearsal scheduling (calendar view)</td>
</tr>
<tr>
<td>6</td>
<td>Networking</td>
</tr>
<tr>
<td>7</td>
<td>Implement the update and sharing functionality for checklists</td>
</tr>
<tr>
<td>8</td>
<td>Extend XML schema to include prop items</td>
</tr>
<tr>
<td>9</td>
<td>Scene and Act Charts</td>
</tr>
<tr>
<td>10</td>
<td>Blocking design tools</td>
</tr>
</tbody>
</table>

Table 10 - Suggested Future work, prioritised by customer needs

All the items identified as recommended future work have been included within Table 10 above; the ordering of items in the table is the suggested ordering that development should occur. Priorities identified during the research phase of the project are not the same; this change is mainly due to the feedback from user testing, but also from decisions made about which requirements would add the greatest value to users. The approach to ranking has been to prioritise those requirements that improve the current experience before adding additional features to the application.

The first item on the list is fixing the bugs and implementing the feature enhancements identified from user testing as included in Table 9, Section 7.2.1. These are relatively small items that fixing would result in quick improvements to the application’s usability and success.

The ability to import contacts directly from the existing iPad contacts was also identified from user testing, but as it would require further work has been separated out. This was a highly sought after feature as if the contacts are pre-existing this would help to reduce the amount of manual typing required to populate details. With only minor modifications to code, this could be implemented using the existing Appcelerator APIs for querying the built in contact database.

The reasons for the inclusion of the tutorial feature have been discussed in detail as part of the analysis of the usability testing results, in section 7.2.1.

The fully featured production application requires that the script be imported in XML format. Future work must provide an efficient means to convert scripts into this format, either as a proprietary service or through customer tools, thus allowing wide use of the application.

The remaining features were all identified within the research phase of the project. As noted above, there has been a change in prioritisation to rank higher improvements over new features. Based on this, the calendar view for the rehearsal is ranked highest in priority for implementation. The calendar view would provide a better for UI for the user when organising rehearsals. Ordering and grouping of rehearsals by date is already included within the feature, but a calendar would provide a more intuitive system.

The next item recommended for implementation is the networking of features. This feature would provide new functionality, as well as improvements; on this basis it has been ranked below the
rehearsal scheduling feature as there would be significantly more effort to implement the features in an effective manner. Appcelerator provide networking services that could be utilised; this would significantly reduce the complexity of implementing this feature.

As part of the networking feature, it will be important to consider how to implement secure transmission and storage of user details and contacts. It will be important to investigate how restricted access to this data could be provided to members of the cast and crew. It is expected that the director would have administrative rights to set up accounts for different users to log into the application. This would provide a simple interface for setting which features were disabled for different users. The security of the personal data was considered in Section 5.3 on ethical issues.

Having implemented the networking, the next logical step would be to extend the functionality of the checklist feature.

As noted in Section 6.1, prop items are not included within the TEI XML schema for performance scripts. The schema should be extended to allow for these items to be queried from the script and provide further functionality for automated lists as part of the checklists feature.

The remaining features would require significant work to implement and would need to be fully investigated to understand demand. It is recommended that user feedback be carried out to understand whether the features would be considered beneficial. When considering the blocking and design feature, there are pre-existing applications that already deliver this with high quality and effectiveness.
9 Conclusion

As a result of the work performed in this project, the author feels the overall objective – to provide a tool to aid the director in the production of a theatre show – has been achieved.

Mandatory requirements were specifically identified for the purposes of the project and the main requirement identified was the integration of scripts into the application through the use of XML tagging. This XML tagging functions as a backbone for other features, allowing for many menial tasks to be automated. The code has been designed in a modular fashion to allow for further requirements to be added as part of further work.

The application has initially been developed for the iOS platform and iPad devices. The iOS platform has a smaller market share than Android devices as a whole; however, the iOS devices are less fragmented which means that less testing was required during the development process to ensure the application was working satisfactorily. The cross-platform development tool, Appcelerator’s Titanium, has been used for the development process to ensure that the application could be easily ported to other platforms and devices.

As noted, script integration was the most important part of the application development, in terms of how its success impacted the other features. The TEI schema was found to be an effective tool for marking up the script and helped to ensure the application was as successful as it was. The integrated script allowed unique features to be implemented that improved the application over the related works.

The Titanium framework was found to be a very powerful tool for development. Whilst there were some initial difficulties in understanding the framework, the end result was found to be responsive and comparable to other iPad applications. Included within the Titanium framework was the Alloy MVC framework and specifically its implementation of Backbone.js. In retrospect, the implementation of Backbone.js within Titanium should have been the influencing factor in choosing SQL storage over NoSQL storage in the requirements analysis as it helped to create such a responsive application.

As well as meeting the base objective by delivering on the identified requirements, the application should be considered a success as it was well received by users and scored very highly in its usability score. The author feels the results of user testing accurately reflect the application’s success. Areas for further work have been identified; however these enhance rather than fundamentally change the application.
References


THE INTERNET ENGINEERING TASK FORCE 2009. Internet Calendaring and Scheduling Core Object Specification (iCalendar).


Appendix 1  
Signed Ethics Approval Form

School of Mathematical and Computer Sciences
Ethics Approval
Student Projects

1. Title of project:
   **Panto Genie: A Theatre Show Production Tool**

2. Supervisor:
   **Rob Pooley**

3. Purpose of study:
   The objective of this project is to produce an application to help in the planning, production and execution of theatre shows.

4. Are human subjects or personal data identifiable with living people involved?
   Yes □
   No X

   If NO sign and date the form, otherwise answer the rest of these questions

5. Is the project mainly concerned with other matters and the only ethical consideration the use of human subjects in interface evaluation?
   Yes □
   No □

   If YES and the subjects are staff of students of Heriot-Watt University, no ethical approval is required, except for the requirement for subjects to sign the standard consent forms and to store these consent forms. Data must be anonymised.

   If NO, then you must complete the full research proposal ethical approval form.
Please sign the following:

I, the student undertaking the project, certify that this is a true reflection of the intended study, and that I will seek the necessary approval if the nature of the project changes.

Name (please print)  **JAMIE HOLWILL**
Signature
Date  **19/08/2014**

I (as the project student’s supervisor) have checked the above for accuracy and am satisfied the information provided is a true reflection of the intended study.

Name (please print)  **Prof Rob Pooley**
Signature
Date  **19/08/2014**

*Figure 30 - Signed Ethics Approval form*
**Appendix 2  Risk Assessment Form**

**MACS Risk Assessment Form (Project)**

*Student:* Jamie Holwill  
*Project Title:* Panto Genie: A Theatre Show Production Tool  
*Supervisor:* Prof. Rob Pooley

**Risks:**

<table>
<thead>
<tr>
<th>Risk</th>
<th>Present (give details) (tick if present)</th>
<th>Control Measures and/or Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Office environment- includes purely software projects</td>
<td>✓</td>
<td>Nothing</td>
</tr>
<tr>
<td>Unusual peripherals e.g. Robot, VR helmet, haptic device, etc.</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Unusual Output e.g. Laser, loud noises, flashing lights etc.</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Other risks</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Appendix 3  Project Planning

Based on the intended development process, the plan was split into the following tasks:

- **Planning**
  - Overall code design, planning and set-up
  - UI High-level design prototype
- **Development**
  - Iterative development process for requirements
- **Testing**
  - User Testing
  - Final Debug
- Dissertation writing (performed in parallel with the above)

The relevant extracts from the project planning Gantt chart have been included below:

![Figure 31 - Overview Gantt chart](image)
Figure 32 - Planning Phase Gantt chart

Figure 33 - Development Phase Gantt chart
### Figure 34 - Testing phase Gantt chart

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing</strong></td>
<td>7.5 days</td>
<td>Mon 21/07/14</td>
<td>Mon 04/08/14</td>
</tr>
<tr>
<td>User questionnaires and feedback</td>
<td>4.38 days</td>
<td>Mon 21/07/14</td>
<td>Mon 28/07/14</td>
</tr>
<tr>
<td>Debug and Finalise</td>
<td>4 days</td>
<td>Mon 28/07/14</td>
<td>Mon 04/08/14</td>
</tr>
</tbody>
</table>
## Appendix 4  Risk Analysis

<table>
<thead>
<tr>
<th>Risk</th>
<th>Probability</th>
<th>Impact</th>
<th>Mitigation Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to properly mark-up script given XML schema or unable to query marked-up scripts</td>
<td>Low</td>
<td>High</td>
<td>The script will be integrated using PDF:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Annotations and comments will still function</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Will not be able to automatically query the script</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Features planned to be automatically populated will require manual set up by the user</td>
</tr>
<tr>
<td>Errors arise in iteration development causing delays before the next iteration can start or Planned hours of working cannot be kept due to other commitments</td>
<td>Medium</td>
<td>Medium</td>
<td>Development is scheduled to complete one month before project deadline. This provides some scope for slippages in time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Holidays may be taken from work to provide at least eight extra hours work time for each day taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Mandatory requirements have been marked in the order of their priority for being completed in the event of not being able to complete all tasks</td>
</tr>
<tr>
<td>Volunteer testers take more than the seven days schedule to test their version of the prototype and provide feedback</td>
<td>Medium</td>
<td>Medium</td>
<td>Volunteers are scheduled to start their testing one month before the project deadline. This will provide some buffer for delays.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Will ensure volunteers know importance of timely feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Chase up volunteers who have not responded by day five of deadline to encourage response within deadline</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- In the even that not all respond in time, there are other evaluation tests been carried out to assess the project.</td>
</tr>
<tr>
<td>Volunteers unable to test application as they do not have access the target platform</td>
<td>Medium</td>
<td>Low</td>
<td>Testing will be made available to users by use of the author’s iPad</td>
</tr>
<tr>
<td>Unable to implement planned features using the Titanium development tools</td>
<td>Low</td>
<td>High</td>
<td>Mitigate by changing development process to native UI application:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- The most complex feature, script integration, has been planned as the first iteration. This will identify any issues early enough to change development process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Forward planning performed will still be relevant for overall design</td>
</tr>
<tr>
<td>Dissertation writing takes longer than planned</td>
<td>Low</td>
<td>High</td>
<td>The dissertation writing is scheduled to be carried out throughout the process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Notes will be taken at every stage for easy reference to the process carried out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Development and testing scheduled to end 11 days before final deadline</td>
</tr>
</tbody>
</table>

*Table 11 - Risk Assessment*
## Appendix 5  User Story Tasks

Table 12 - User tasks for usability testing, including information on features tested

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Feature</th>
<th>&quot;Story&quot; order</th>
<th>&quot;Story Based&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL.3</td>
<td>Remove a contact</td>
<td>Contact - remove</td>
<td>1</td>
<td>Gordon, who was playing Puss originally has now left the show. Remove him from the contacts.</td>
</tr>
<tr>
<td>CL.1</td>
<td>Add a new contact</td>
<td>Contact - add new</td>
<td>2</td>
<td>You have just given the part of Puss to Jon, create a new contact for him with the following details….</td>
</tr>
<tr>
<td>CL.2</td>
<td>Edit a contact</td>
<td>Contact - edit</td>
<td>3</td>
<td>Eugene, the Stage Manager, has changed her email. Edit her entry for the new email &quot;<a href="mailto:eugene@pantomagic.net">eugene@pantomagic.net</a>&quot;</td>
</tr>
<tr>
<td>R.1</td>
<td>Schedule a rehearsal for tomorrow at 12:00pm for 2 hours. Scene 2.</td>
<td>Rehearsal - create new rehearsal</td>
<td>4</td>
<td>You need to organise a rehearsal for Act 2, Scene 3 tomorrow at 12pm for 2 hours. Set this up now.</td>
</tr>
<tr>
<td>R.2</td>
<td>Invite a non-intended person and remove someone too.</td>
<td>Rehearsal - contact edit</td>
<td>5</td>
<td>You also want the stage manager to come to this rehearsal, so add her to the invite list. You know Craig, playing Jack, can't come to the rehearsal. Remove him from the contact list.</td>
</tr>
<tr>
<td>R.3</td>
<td>Send the email rehearsal</td>
<td>Rehearsal - email feature</td>
<td>6</td>
<td>Save and send the rehearsal email</td>
</tr>
<tr>
<td>R.4</td>
<td>Edit a schedule rehearsal</td>
<td>Rehearsal - edit feature</td>
<td>7</td>
<td>You get a response from John that he can't come till 3pm - edit the rehearsal to reflect this change.</td>
</tr>
<tr>
<td>CK.1</td>
<td>Create a new template checklist for a rehearsal</td>
<td>Checklist - template</td>
<td>8</td>
<td>Before the rehearsal, you should set up a checklist for a rehearsal. Set one up using the rehearsal template.</td>
</tr>
<tr>
<td>CK.2</td>
<td>Create a new task and add it to the 15 minutes before start</td>
<td>Checklist - add new item (inc. sections)</td>
<td>9</td>
<td>You know that you need to get props out of storage for the rehearsal - add a new item to the checklist for before the rehearsal starts, for this task.</td>
</tr>
<tr>
<td><strong>CK.3</strong></td>
<td>Delete an item from the last section</td>
<td>Checklist - item edit</td>
<td>10</td>
<td>You have run out of budget for rehearsal snacks, remove these items from the checklist</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------</td>
<td>----------------------</td>
<td>----</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>S.1</strong></td>
<td>Navigate to Act 2, Scene 3</td>
<td>Navigation through script - contents</td>
<td>11</td>
<td>The scene being rehearsed will be Act 2, Scene 3 - navigate to this in the script view</td>
</tr>
<tr>
<td><strong>S.2</strong></td>
<td>Highlight Jack's lines</td>
<td>Parts highlight</td>
<td>12</td>
<td>You will be playing the part of Jack - highlight your lines.</td>
</tr>
<tr>
<td><strong>S.3</strong></td>
<td>Swipe 3 pages to the right</td>
<td>Navigation through script - swipe</td>
<td>13</td>
<td>You are 3 pages into Act 2, scene 3 during the rehearsal, navigate to this page</td>
</tr>
<tr>
<td><strong>S.4</strong></td>
<td>Add a comment to the script</td>
<td>Comments - freehand</td>
<td>14</td>
<td>On the third page, you note that on &quot;You!&quot; that Jack should really emphasise this line. Add a comment to &quot;Shout this line&quot;</td>
</tr>
<tr>
<td><strong>S.5</strong></td>
<td>Add a template comment to the script</td>
<td></td>
<td>15</td>
<td>You think Nosmo should point Stage left when saying &quot;We took them to the Ogre's Castle&quot; - add a comment to this line, with the help of the template comments, to take a note of this.</td>
</tr>
<tr>
<td><strong>CK.4</strong></td>
<td>Check off items on the checklist</td>
<td>Checklist - checking items</td>
<td>16</td>
<td>It's the end of the rehearsal, check off the item for locking doors</td>
</tr>
<tr>
<td><strong>T.1</strong></td>
<td>Set up a new timer group for a performance</td>
<td>Timer - templates</td>
<td>17</td>
<td>Its show time! Set up a timer group, using the templates, for a performance.</td>
</tr>
<tr>
<td><strong>T.2</strong></td>
<td>Add a new timer to the template, &quot;Post-production stage clean up&quot;</td>
<td>Timer - new timer</td>
<td>18</td>
<td>You also want to time how long it takes to clear up after a show. Add a new timer for this.</td>
</tr>
<tr>
<td><strong>T.3</strong></td>
<td>Timer - start / stop a timer</td>
<td>Timer - timer feature</td>
<td>19</td>
<td>The show has started, start the timer for the show....</td>
</tr>
</tbody>
</table>
### Appendix 6  Usability Testing Results

#### Table 13 - Individual and Aggregated SUS scores

<table>
<thead>
<tr>
<th>Question</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
<th>User 6</th>
<th>User 7</th>
<th>User 8</th>
<th>User 9</th>
<th>User 10</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adjusted Total scores</th>
<th>34</th>
<th>23</th>
<th>34</th>
<th>36</th>
<th>37</th>
<th>34</th>
<th>32</th>
<th>34</th>
<th>20</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUS Score</td>
<td>85</td>
<td>58</td>
<td>85</td>
<td>90</td>
<td>93</td>
<td>85</td>
<td>80</td>
<td>85</td>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>Total time</td>
<td>15:57:00</td>
<td>24:46:00</td>
<td>19:32:00</td>
<td>10:48:00</td>
<td>8:40:00</td>
<td>10:52:00</td>
<td>10:43:00</td>
<td>12:55:00</td>
<td>27:58:00</td>
<td>9:46:00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average</th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time</td>
<td>15:11:42</td>
</tr>
</tbody>
</table>
### Table 14 - Unadjusted SUS scores

<table>
<thead>
<tr>
<th>Responses Provided</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Question</strong></td>
<td>**User 1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>