

Group 10 Final Report

February 17, 2014

SE.10.FINAL.1

Version: 1.0
Status: Release

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager & Web Developer
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2014

Contents

1	INTRODUCTION	3
1.1	Purpose of this Document	3
1.2	Scope	3
1.3	Objectives	3
1.4	GitHub Repository Structure	3
2	End-of-Project Report	4
2.1	Completion of Features	4
2.2	Management Summary	4
2.3	Historical Account of the Project	4
2.4	Final State of the Project	6
2.5	Performance of each team member	7
2.5.1	Daniel Clark - Group Leader	7
2.5.2	Charlie Newey - Assistant Group Leader and Lead Android Developer	7
2.5.3	Stephen Mcfarlane - Lead Web Developer	8
2.5.4	Mark Lewis - QA Manager and Web Developer	8
2.5.5	Martin Ferris - Android Developer	8
2.5.6	Ashley Iles - Android Developer	9
2.5.7	Kenny Packer - Android Developer	9
2.5.8	Kieran Palmer - Web Developer	10
2.5.9	Critical evaluation of the team and the project	10
3	APPENDICES	11
3.1	Reflective Reports	12
3.1.1	Daniel Clark	12
3.1.2	Charles Newey	13
3.1.3	Stephen McFarlane	14
3.1.4	Mark Lewis	15
3.1.5	Martin Ferris	16
3.1.6	Ashley Iles	17
3.1.7	Kenny Packer	18
3.1.8	Kieran Palmer	19
3.2	Maintenance Manual	20
3.3	Test Log	35
3.4	Revised Test Specification	43
3.5	Revised Project Plan	52
3.6	Revised Design Specification	75
4	REFERENCES	118
5	VERSION HISTORY	118

1 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to give both a brief overview and an in-depth report into the quality of the finished project. It is also to give a maintenance manual for both sides of the project to ensure future development of the project is as simple as possible.

1.2 Scope

This document will cover a Management Summary which is a brief overview of various elements of the entire project, including it's state of completion and how the team worked to get it to its current state. It also includes a Historical Account of the Project which entails a step by step overview of the major milestones in the project in order from oldest to newest. A section for the Final State of the Project is also included which goes into more detail about exactly what parts of the software works and what doesn't.

An individual assessment of the performance of each Team Member is also included, to go into detail about the contribution and progress every member of the team has had over the course of the project. Lastly there is a brief critical evaluation of the team and the project which explains what we have learnt from the project and what we would change for next time.

It will also contain a Maintenance Manual for both the Android and the Web sides of the project to ensure that the future development of the projects source code is as simple as possible. This document will also supply the management with a personal reflective report from every member of the group on how they think the experience went, plus the revised versions of the previously submitted documentation after the recommended changes have been made.

1.3 Objectives

The objectives of this document are to:

- Give a suitable summary for submission to management to give an idea of the success of the project.
- Give an in-depth view of the projects progress for the Project Manager to use to judge the quality of the software and documents produced in the project.
- Give future developers a set of guides and materials for easy continued development of all software in the project.
- Give a personal reflective report for each group member to give the Group Manager a good idea of how the project was run and how each group member felt the project went.
- Give up-to-date versions of the previously submitted versions of documents, including changes that the Project Manager and customer had suggested.

1.4 GitHub Repository Structure

As a point of information, the project is stored in two repositories, so as to avoid mixing code bases (this is commonly regarded as bad Git practice) - and this also allowed us to keep our repositories under GitHub's maximum recommended size of 50mb. This also made it significantly easier to use the Android project in Android Studio with the Git integration.

The URLs of the repositories are:

- Document and Web Repository - <http://github.com/Danyc0/CSG10.Git>
- Android Repository - <http://github.com/charlienewey/CSG10.Git>

2 End-of-Project Report

2.1 Completion of Features

As mentioned in the report below, the application is almost feature-complete to the customer's standards. However, during the planning phase of the project, extended functionality was decided upon and designed. This was later decided to be unnecessary and surplus to the customer's requirements - and due to time constraints, we did not implement these extra features as they were a result of feature creep, and potentially would have been counterproductive.

2.2 Management Summary

The project as a whole has achieved almost everything it set out to do. The app is in full working order and is well tested with only a few minor features missing; and these can easily be added at a later date.

One missing feature is that it is not possible to add multiple images to each Waypoint. This is because we misunderstood the specification until the last day of Integration and Testing Week, at which time we had no time to change it (this is because we thought it meant that you needed multiple photos per route, not per waypoint). The other missing feature is that the user is unable to edit Route metadata once it has been created. The website also works perfectly and has every feature specified in the requirements specification plus the added feature of a gallery at the bottom of every photo in a route.

All submitted documents were submitted on time and to appropriate standards regarding the Quality Assurance standards that have been put in place. All documents were checked for errors and were checked to ensure they contained the correct material for submission. After feedback was received on the initial document versions, changes were immediately implemented ready for final submission.

The main difficulties experienced during the completion of this project were illness from group members, and the adverse affects the weather had on the number of workable days for the project. Over the course of the project around half of the group have missed meetings due to illness and although all work was later caught up on, full attendance of all meetings would have ensured higher efficiency throughout the project. The impact the weather had on the projects completion has been small, but only thanks to the work done to make up for missed time by group members.

As a whole the team performed fantastically, in general most team members pulled their weight and produced work of a good quality resulting in a well rounded and complete submission. A few team members throughout the course of the project did not however contribute to the submission as much as the rest of the group, which ended up resulting in them receiving a yellow card. A few other minor problems arose during the course of the project but they were dealt with quickly and didn't end up impacting the final quality of the submitted project.

2.3 Historical Account of the Project

The first thing the group did, with the assistance of our project manager, was decide on group roles - setting out who would be doing what based on their skills and experiences. We decided that Daniel would be the Group Leader because he had experience with leading a group as well as having extra time due to having only 50 credits in the first semester. The Assistant Group leader was decided to be Charlie because he was also enthusiastic about leading and was an experienced Java programmer, leading him to also be the Lead Android Developer. Stephen volunteered then to be the lead Web Developer as he enjoyed working on web design and he was doing the web modules. The rest of the group remember were then divided up into doing either Android or Web, based on where their skill set lied and which they preferred, except Mark who became the Quality Assurance (QA) manager.

Next we started to decide which parts of the initial documentation submission (the Project Plan) would be delegated to who. This resulted in most of the group receiving sections of the Project Plan to complete,

whilst two of the group (Mark and Martin) were told to start work on writing out the Test Specification for the next deadline. After the Project Plan submission was completed, the two who had been working on the Test Specification continued to work on that whilst the rest of the group started working on the first stages of the Design Specification. When then the results for the Project Plan submission were released Mark was given the task to making the initial changes, but only for things that could be done quickly, leaving the harder things to be changed later in the project.

The Test Specification was then submitted after being Quality Assurance checked and some extra tests added after it was looked over by other members of the group. At this point the Design Specification was coming along nicely and now the two who were focussed on the Test Specification now joined the rest of the group in working different sections of the Design Specification.

After a few group meetings involving making some of the larger design decisions we eventually had a Design Specification that was ready for submission and so after being QA checked, it was submitted on time. This meant that everybody could now start coding and working towards a working prototype for the Prototype Demo. Stephen and Kieran were set to work together on making a statically generated version of the web-site to demonstrate what it would look like, whilst the rest of the group were given individual parts of the Android App to focus on and get working, e.g. Daniel was to work on getting Post Requests to his server working using JSON, Charlie was to allow the taking of photos, etc. The plan was then to merge all the working parts of the App together.

When the time came for the Prototype to happen Stephen had finished the static website on his own, with minimal help from Kieran and most sections of the Android app were complete but the parts had not been merged together. The demonstration went well with the 'customer' commenting that no other group had got HTTP Posting working other than us at that stage. It was at this stage that it was decided to give Kieran his Yellow Card as up until this point he had not contributed anywhere near as much as others in the group.

After that we got the feedback on the Design Specification and Mark started work on making the changes needed whilst the rest of the group started on the finished products for both web and Android. After Mark had made the initial changes it was at this point that he volunteered to be in charge of the database interaction side of the web system, as using SQL and doing database work was relatively easy for him. This was also because he had noticed that the web side had not been progressing at the same rate as the Android development. Over the Christmas holidays not much work happened on the web side - some slight UI changes and not much else, whereas on the Android side each element demonstrated in the prototype had finally been merged into one united app.

At the start of Integration and Testing week the Android app seemed very close to completion but the Web was still lagging behind, with still no facility for receiving HTTP POST requests. Throughout ITW Charlie, Martin and Daniel were focussing on adding features and fixing bugs in the Android code whereas Ashley was working on the User Interface and Kenny was doing continuous testing on any new features to check for any bugs that got introduced during development. On the web side Stephen was working on getting the Website to be generated using data from the Database (with some help from Charlie at the end) whilst Mark worked on developing the Database and adding data to it from the HTTP Posts. Kieran was also given the task of redesigning the web interface and updating the docs, specifically the Test Specification so that we were clear which tests we passed and which tests we still needed to work on passing.

After ITW the entire group was working on documentation, most of the group was making the recommended changes to the previously submitted Docs whilst Daniel wrote the End of Project Report and Charlie compiled each section into one LaTeX document, doing Quality Assurance checking as he went. We realised at this point that a lot of the previous changes Kieran had made to the documentation were sub-satisfactory and therefore it was decided not to revoke his Yellow Card.

2.4 Final State of the Project

Both sides of the delivered software package work almost perfectly to meet the delivered specifications as set out in the project documentation. The app contains functionality to create a walking tour containing the title, short description and long description for the route, as well as a title, description and photo for as many waypoints as the end user wants to create. The only point at which this strays from the set specification is we have not yet implemented the limit to restrict only single word titles for each walk.

The GPS coordinate for the tour are started gathering from the point at which the user finishes entering the general route details, then an extra set of coordinates are attached to each waypoint created as part of the tour along with its Unix Epoch timestamp. When it comes to adding photos to the created waypoints, the user is prompted whether they want to take a photo from their camera or add a photo which is stored in their phones gallery as per the specification, however due to an initial misunderstanding of the documentation only one photo is able to be added to each waypoint.

The route is able to be cancelled at any time by pushing the 'back' button on their phone, which will then prompt them to whether or not they truly wish to cancel recording the route. Data sent to the server after final completion of each route is sent via an HTTP POST request containing a set of JSON key-value pairs with a MIME-Type of "application/json", the JSON format was chosen over the MIME format because the existing libraries for Android and PHP made the specifics of coding the POST request far simpler as well as JSON having the extra benefit of being more human readable than just standard MIME. We informed the customer (Bernie Tiddeman) of this change during the design phase of development and he accepted the change.

When the user decides to switch android applications the app will cache the stored data and reload it when the user restarts the app. This process will also handle when the users exits the app for long periods of time and the Android OS kills the App to save RAM. A bug we found in this part of the feature through testing though is that the list of data is not refreshed until you add a new waypoint, however the data IS there either way and can be sent to the server. The apps service will also continue to run in the background recording the GPS track even if the app does not have focus, meaning the user can continue to use the phone for other things whilst walking between waypoints and then restart the app when they decide to make a new waypoint.

On the web side of the software package the user is able to pick any uploaded route from the drop down box and all the information about that route will be displayed to the user, the details of the route itself are displayed at the top of the page, whilst the details of each waypoint are stored within markers on an interactive map in the centre of the page.

Once data has been successfully received by the web server it will return a standard HTTP 200 code confirming that it received the data, or another code depending on what problem was encountered in the transmission. This will then be checked on the android app and an appropriate message will be displayed to the user regarding the success of the delivery, and if successful, they will be given the opportunity to start a new route.

2.5 Performance of each team member

2.5.1 Daniel Clark - Group Leader

As a whole the duties taken on by the group leader were the organisation of the project, delegation of tasks, making sure that the team were happy and comfortable with their assigned work as well as ensuring that the project was progressing at the desired rate.

The group leader's attendance to all whole group meetings was ensured, however some group meetings only required attendance by each side of the programming team and thus the group leader did not always attend. Throughout Integration and Testing Week the group leader was present every day from 9-6 and made sure to supervise any problems the group was having and dealt with any problems involving members of the group that were not arriving on time. Throughout the project the group leader has also input significant amounts of functional code into the Android development, as well as taking a large role in debugging and solving problems found by other members of the team on both the Android and PHP sides of development.

Throughout the project a significant degree of enthusiasm for the project has been maintained, however, post-Integration and Testing Week enthusiasm for the project started to lower as the deadline approached. In general work produced by the group leader was to the standard expected by the rest of the group and all duties were taken care of appropriately. That said, the efforts of the Assistant Group Leader (Charlie Newey) to keep the project on course has compensated for some organisational shortfalls of the project leader throughout the later half of the project.

Acceptance: I (the group leader) as writer of this document, agree with the above self-appraisal.

2.5.2 Charlie Newey - Assistant Group Leader and Lead Android Developer

The duties taken on by Charlie extend far further than the limits set out by his official project roles. Officially his role entails a lot of the group leaders responsibilities but to a lesser degree, as well as taking charge of the Android development and making sure that everyone within the Android coding team knew exactly what needed doing and which tasks they were assigned to. On top of maintaining a high quality of work in his official roles he also continued to support the group leader and compensate for any shortcomings he came across throughout the project. On top of the tasks he was assigned he has also made contributions to the Quality Assurance of both the documents and all parts of the code and spent a not-insignificant amount of time working with the team's Web developers to fix bugs and add features to that side of the project in any free time he had during Integration and Testing Week.

Charlie made attendance to almost every meeting of both the general group and the Android development meetings although he did sometimes turn up late. During Integration and Testing Week he consistently arrived an hour after the start to the day, but he always made up for this by spending extra hours in the evening working on the project. Charlie took one day at home during the week (but continued to do work) as he was expecting a large delivery - this was out of his hands as it had originally been organised to not clash with ITW, but the adverse affects of the weather delayed ITW by a week, thus causing the problem.

Charlie's enthusiasm for the project has been high all throughout the project often spending extra hours perfecting sections of work he and others had produced. His high enthusiasm seems to be driven by the desire for the entire group to receive over 70% for the project and he went out of his way to try and achieve this. The quality of work produced by Charlie throughout the project has been strong, producing readable and complete sections for documentation when required, and producing a high quality of well documented code during Integration and Testing Week.

Acceptance: I, Charlie, agree with the above statements. I made a large effort to restore any project time lost, and did so several times over. I feel, however, that my contribution to the the project was a little higher than my 'fair share', and that I had to pick up slack left by some team members. With that said, I would do so again if it ensured the project reached successful completion. The comments on my attendance during ITW are fair, but reflect the extremely late nights I was working on the project - long after the day had officially ended.

2.5.3 Stephen Mcfarlane - Lead Web Developer

Stephen, as he had experience of PHP, was set the duties of writing the majority of the projects web side. He alone produced a working static version of the website for the prototype submission and during Integration and Testing Week adapted it to work with dynamic data pulled out of the database. Stephen made it to all scheduled meetings before and after Integration and Testing Week, however during the early part of ITW he was at least 2 hours late on most days. Despite being late on some days during ITW, he always made the effort to make up work he'd missed during the evening so that his tasks were completed and up to date before the next morning, meaning that the project was not held back. When it comes to Stephen's attitude towards the project, he has continuously proven to be a consistent hard worker, every task set to him was completed on time and it was obvious to see the progress he was making as time progressed.

The code produced by Stephen for dynamically delivering the website to the user was completed to a good quality, however the JavaScript which was used in conjunction with the Google Maps API to display the route to the user could have been written in a tidier and more readable way. Towards the end of ITW Stephen was struggling with getting the last part of the website working, but after a code refactor and some help from Charlie, a solution was found. The parts of the documentation he wrote to go along with the project were complete and well written to a good standard. In general Stephen is very easy to work with and will always complete whatever he is given to the best of his ability.

Acceptance: I, Stephen, have read this and approve this report.

2.5.4 Mark Lewis - QA Manager and Web Developer

Mark's original role was solely to be the QA Manager, a job that during the pre-Integration and Testing Week stages he did to a good standard - making sure that documents were completed to the correct standard before submission. When the group started working on getting the final code working however, Mark offered to lend a hand with the Web Development as he had some experience with working with SQL and the database technology.

Mark made it to almost every meeting and took up the general role of writing the meeting minutes. If he was ever unable to attend, he notified me beforehand and work was set remotely. During ITW Mark was often the first to arrive just before 9am and stayed until the end of the day. When Mark realised that he could use his skills to benefit the group by working alongside the other Web Developers and volunteering himself for the extra role it was obvious that he was motivated to make sure the project was a success and was completed on time.

In his role as QA manager Mark also spent time at the end of Integration and Testing Week improving the quality of the back-end web code to ensure it met the coding standards that had be set out. Over every aspect of the project that Mark worked on he made sure to produce a quality piece of work. Even from the start, the minutes of each meeting were taken professionally without prior instruction, documents were QA checked before submission on time and his code was well written and well commented despite needing some occasional help from other group members.

Acceptance: Mark agrees with Dan's comments.

2.5.5 Martin Ferris - Android Developer

Martin's role as an Android Developer throughout the project has led to him producing a significant amount of the code for the Android App, as well as spending time at the end of the project finalising the document changes. Throughout the project Martin's attendance has been fantastic, he turned up to virtually every organised group meeting and made it to every day of Integration and Testing Week. The only problem in Martin's attendance was that he very rarely showed up on time for ITW, however that did get better as the week went on and he always made up for missed time in the evenings.

He has had a high level of enthusiasm throughout the project consistently finishing work and asking what needs to be done to improve the final product. Martin took the responsibility of writing the sets of JavaDoc

comments for almost all of the Android code and got it ready for generating the final HTML JavaDoc. He also did work on building the UI for some of the screens working with Android Studio and directly with the XML. He is often willing to go the extra mile in ensuring documents are completed on time for a deadline and that feedback from the customer is appropriately dealt with.

The quality of work the Martin produces is consistently of an acceptable standard in both code and documentation. Although at times he struggled in fixing bugs and implementation problems in the code, but with minimal guidance he eventually worked through every problem. Generally Martin is very easy to work with and, with the right support, will produce quality solutions to problems.

Acceptance: I, Martin, have read this report and approve of Daniel's comments.

2.5.6 Ashley Iles - Android Developer

The role set out for Ashley initially was to be one of the Android Developers and out of everyone he stuck to his role the most.

His attendance through all of the official meetings was generally very good, however he missed some of the non-official meetings. This was never due to him being unenthusiastic, it was because the meetings were always organised via Facebook and Ashley has never been a strong Facebook user. Ashley's attendance during Integration and Testing week was not fantastic, often turning up late and on one day he didn't turn up at all, but that was due to illness (although we were only told this the day after). Despite this however he did partially make up for that by staying late one one night to make some final changes.

Pre-ITW Ashley did work on the designs for the User Interface of the Android app including the initial designs and the initial way the user would use the App in the Sequence Diagram. During Integration and Testing Week Ashley ended up mainly focussing on the User Interface design for the Android App, as he generally has a good eye for what looks good. This involved things like working with the UI tool in Android Studio as well as directly editing the XML when he needed to. It also involved a reasonable amount of editing images that Charlie had taken previously for use as the backgrounds for the App. He also did work with the timestamps in each waypoint and was generally considered the go-to guy whenever the XML was involved.

Ashley's quality of work was generally good, however at times it required quite a lot of effort to get work out of him in the first place.

Acceptance: I, Ashley Iles, do agree that the above is an accurate description of my involvement in this group project.

2.5.7 Kenny Packer - Android Developer

Kenny was assigned to be part of the Android Development team. Throughout the project Kenny spent most of his time working on developing the Android app as well as writing the initial versions of some of the documentation and making the changes as recommended by the customer.

Due to timetabling commitments it has meant that he has had to miss some of the meetings planned with short notice, but he is always happy to meet after the meeting to discuss what needs doing. During Integration and Testing Week Kenny showed up every day and continued to work on the App with the rest of the group, he also took the lead in testing the app after each new feature was marked as complete, often flagging up major bugs which needed fixing. Kenny also headed up making the more detailed visual design decisions for the app, e.g. designing the apps logo.

Throughout the project Kenny would always be up for taking on any task set and was not afraid to ask for help when he needed it. He often seemed to feel more comfortable working in tandem with someone else on a task rather than working completely on his own. He also was often used as a method of coordinating with Kieran when he was not available by other means or in meetings.

The quality of Kenny's work stayed consistent all the way through the project, always completing documentation as needed which generally only needed minor spelling and grammar checking. Although Kenny did not contribute a significant amount of code, the code he did produce was of high quality. On the final day of the project Kenny did a fantastic job on checking the final document and finding numerous mistakes which were able to be corrected before submission.

Acceptance: Pretty much true, no issues with this evaluation - Kenny

2.5.8 Kieran Palmer - Web Developer

Kieran's role in the project was set out as a Web Developer, whose role was mainly to support and contribute to the web front-end side of the project. He also took charge of making the initial changes to documents once they were returned after the first submission. Throughout the project Kieran has missed a significant amount of group meetings - both official timetabled meetings and individual group meetings. Kieran also missed a few days of Integration and Testing Week and on the days he did attend, was often late.

In the first part of the project Kieran mainly focussed on documentation of the web side, producing parts of the document like the web side Sequence Diagram. During Integration and Testing Week Kieran's main focus for the project was on improving the UI for the web design. At the start of the project it seemed that despite his initial enthusiasm, the progress made by Stephen left Kieran without the opportunity to contribute at that stage of the project. Kieran's general attitude towards the project has been reasonably good whilst in meetings, however between meetings his contributions to the project seemed to dwindle. Towards the end of the project Kieran's attitude seemed to improve, often asking what extra work he can do and what he could contribute to.

The quality of documentation produced was generally good with normally only minor corrections needed. Kieran has struggled throughout the project on learning to use the Git version control, always needing supervision from other team members before committing any progress he makes.

I think with more managed time and meetings Kieran could be a much more valued member of the team.

Acceptance: I, Kieran, agree with what Daniel has written about my efforts in the group project. I acknowledge that my contributions were poor and I appreciate the help given to me by other members of the team.

2.5.9 Critical evaluation of the team and the project

As a whole, the team performed fantastically, which resulted with a finished product I think we can all be proud of producing. The contributions of some team members significantly outweighed the contribution of others, which led to an unbalanced separation of the work. However on average the team pulled together and balanced out all the work that needed doing.

I think the project as a whole could have been improved by spending more time focussing on making sure the Git version control was nailed down and fully understood in everyone's mind, because it allows for far greater collaboration even when group members aren't in the same room. Another improvement I'd make if I did this again would have a better adopted communications system due to the fact that neither Kieran nor Ashley spend much time on Facebook, meaning they missed out on a lot of the communications that were happening.

As a group, we all agree that the most important lesson learned about working on software projects is that communication is key, everybody needs to know exactly what's happening, when it's happening and what they're meant to be doing. Another important lesson is about making sure that the less able people involved in a project feel like they are making a worthwhile contribution to the project to make sure that their enthusiasm stays high and they produce work to the best of their ability. Something else we've learned that will help us in future group projects is the value of having dummy systems when working on two platforms, e.g. we had a dummy web server running that we used to test that the Android app could send data long before the actual web server was functional. Furthermore, the way we structured this project is proof to us

that good planning, good testing and incremental development really do help to produce a quality product at the end.

3 APPENDICES

Over the next few pages, various appendix items are included:

- Reflective reports from all team members.
- A test log - which should provide sufficient evidence of testing and the functioning state of the application.
- Revised and updated versions of the Project Plan and Design Specification
- A full maintenance manual to aid any future software engineers in solving problems and improving the project

This should hopefully give a full overview of the project's structure and measured success.

This space is intentionally blank

3.1 Reflective Reports

3.1.1 Daniel Clark

Daniel's Personal Reflective Report

=====

I think on balance the project went extremely well. As with other groups it took us some time to get to know each other and learn each others strengths and weaknesses, but as time progressed the speed and efficiency at which we worked greatly increased. One of the major problems we had in the group to start was that, having never worked together before, we had no idea what each other considered as 'good', and what we all expected of each other. As the project progressed it was obvious that some group members were more motivated to succeed than others, but in general there was definitely a positive attitude toward the group throughout. During Integration and Testing week the group really started to become a unit, where everybody knew what they were doing and always knew what the next step to take was.

As an individual I really enjoyed the project, learning how to use development tools and techniques effectively, learning about the Android development environment and how powerful the Android API is. Also learning about PHP, as language I've never even touched before, was extremely interesting and has given me a much better background in the topic for further progression. As Group leader I feel that I did a good job leading the team, maybe at times not the best but I still think I've managed to lead the team into getting the desired result. Leading a team with such varied abilities has taught me an awful lot about how important good communication is, and how it's important to make sure that each member of the team feels like they are making a valuable contribution to the project. I also learnt that being liked by your group is extremely important in keeping up group moral throughout long projects, as without that the entire group ethos can fall down.

Although I did end up spending massive amounts of time on the project, I always expected this and that is one of the reasons I put myself forward as group leader. This is because I was only taking 50 credits in the first semester which left me much more time to focus on the project and its development.

If I were to ever repeat the project or one of a similar nature, I'd have made sure I put more effort into making sure every team member got the support they needed and felt that they could ask for extra help/resources if needed. Also, although I tried to enforce good Git practice right from the get-go, next time I would definitely focus more on making sure every single team member was completely happy with using it's basic operations and how to deal with conflicts.

Total time spent on project

Estimated project work time 130-140 hours.

3.1.2 Charles Newey

Personal Reflective Report

=====

Overall, I'd feel the project was successful. Despite being slow to start, the group began to work more effectively as a team during Integration and Testing Week, to a point that our "turnaround time" for bug fixes was less than an hour. The majority of the app functionality was implemented and the vast majority of tests passed.

The major difficulty of the project was the process of getting to know the team's abilities initially, it was difficult learning everybody's strengths and weaknesses. We worked together as a whole, although I feel perhaps that I did more work than was my 'fair share'. This was due to perhaps underestimating the amount of work that would be involved in the project, and the learning curve for the project tools (Git, GitHub, and LaTeX) all presented a challenge for the team and was a recurring difficulty throughout the project, requiring help from myself and Daniel.

We also faced a number of problems with group members, including late submissions of work from some group members (for internal deliverables), poor attendance, and extremely poor communication. This led to a number of complications that I feel myself and Daniel have had to take on disproportionate workloads to compensate for. In all, I did a very large portion of the overall work, and this has been a major frustration to me during the course of the project.

Despite this, I enjoyed the project. The Android development was interesting, as well as complementing my own skill set as a developer. Working on a code base with a small team was a new experience for me - one that I enjoyed overall. Working in a team enabled better discovery of flaws, faster turnaround time on bug fixes, and more efficient code as we held informal code reviews often during Integration and Testing Week. Outside of ITW, the team were generally good to work with, although the difficulties mentioned above did somewhat sour the experience.

If I were to repeat the project - personally, I'd have learned a little more about Git and GitHub before the project started. Learning about Git is one thing, but it's very different to use in a group setting, and can be quite a steep learning curve for a new user. This became less of a problem as the project progressed, and I was able to help others - but this was initially a problem as it would have been more useful to be able to help some of the other team members with learning project tools - instead of spending time learning them myself.

Total time spent on project

Estimated at 135 hours. (Summary of time in blogs + 10 hours for final document)

3.1.3 Stephen McFarlane

Personal reflective report

=====

Well I think the group project went great the group worked well as part of a team and we got the project working on both web and app side. I learnt a decent amount about how to use google maps when creating a web page which means I learnt more javascript, also with the database interaction I had to do some PHP, SQL and XML. The only thing I would have changed was learning how to use Git before the project started as it was difficult to learn while doing and made some minor errors.

We had a few problems with the website like not having a database up and running for a while and the database interaction with the Google maps JavaScript, but this was all sorted out and was finally fixed.

Personally I found the project enjoyable I learnt alot and was fun at some times.

total amount of hours on project : 60-65 hours over pieces of work and 30 hours looking up information to create website such as researching Google maps and php techniques to get the website to work.

Total hours : 90-95

3.1.4 Mark Lewis

Mark Lewis - 31/01/14

=====

Role: QA Manager/Web developer

Review

I think that the project went very well. The start of the term involved alot of documentation which we completed and got good grades on. Then when it came to producing the system I felt it went better then I expected it to go. I was mainly involved with the web side of the system, there were a few problems which we encountered and at the beginning I didn't expect everything to go smoothly. But these were all fixed and all the requirements where met.

Overall I probably spent between 90-100 hours on the project.

3.1.5 Martin Ferris

Martin Ferris - 31/01/14

=====

Role: Android Developer

Personal Reflective Report

Overall, I think the project went well. There was really good communication throughout amongst the group, and I feel that we produced a final product to a high standard. At the beginning of the project I was a bit nervous about being a developer, but everyone was supportive and I never faced a problem that I could not solve even if it did take some time to get through.

I really enjoyed learning how to develop android programs. It was challenging but rewarding when we succeeded and reached our goals. I feel we had the right approach by having lots of short term goals, opposed to trying to complete a whole segment of the program in one go.

If I were to repeat the project I would have liked to have been taught Git in a more practical way, but thankfully my group members had some Git experience. Also, I would have used the advisory service more as I wasted a lot of time on bugs.

Total time spent on project

Estimated at 120 hours.

3.1.6 Ashley Iles

Ashley Iles: Android Dev Team

=====

Overall I do believe that the group that I was part of was quite successful, the final product are two very functional and easy to use pieces of software. My previous, but limited, experience / knowledge of Git and Github was very useful during the project. This resulted in myself requiring help with Git to be a very infrequent event. When refining the UI design in android studio that either the other group members or myself had created I found myself frequently editing the XML files directly. Cleaning up the mess and unnecessary attributes that the IDE had created and left behind was quite satisfying.

That said I found myself feeling underwhelmed and rather unmotivated, especially compared to others in my group. This feeling did change during the testing week, working in close proximity to my other group members was something I found very beneficial.

Estimated time spent on project ~70 hours.

3.1.7 Kenny Packer

Kenny Packer - 31.01.2014

=====

Role: Android Dev Team

Its done, more or less, its done. And it works really well.

I'm glad the app works really well. I'm also glad that I learnt some new things, like being able to use GitHub (sort of). I also got to create the mascot/icon for our app, write some code, field test the app, point out errors, and just generally being useful to my group. I'm not the strongest programmer, or computer science student in general, so I felt I had to be useful wherever I could to compensate for this. The fact that we had a really good leader and deputy, meant that I could easily ask for help with any project stuff, and that meant I could get a good amount of work done. I improved with computer science stuff, which is good. The fact that we managed to create something useful (in theory), from basically nothing but an IDE and some programming know how. In a general sense, Its renewed my faith in programming in Java.

Total time spent on project

40+ hours

3.1.8 Kieran Palmer

Personal Reflective Report

I feel that the project was overall enjoyable and I learned a considerable amount. However I do feel that I did not apply or commit myself as much as I could, but realising that is positive. I have gained experience from this project and I have grown academically and personally. The main aspect that I appreciate that it was a productive break from learning and being able to apply what I have already learned was refreshing.

There were issues during construction of the software but over coming them in teams was very helpful in gaining usable experience.

Total time spent on project

Estimated at 40 hours

Group 10 Maintenance Manual

February 17, 2014

SE.10.MAN.1

Version: 1.0

Status: Release

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager & Web Developer
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2014

Contents

1 INTRODUCTION	4
1.1 Purpose of This Document	4
1.2 Scope	4
1.3 Objectives	4
2 Android Maintenance Manual	5
2.1 Program information	5
2.1.1 Program Description	5
2.1.2 Data flow diagram	5
2.2 Data Classes	6
2.2.1 StumblrData	6
2.2.1.1 Significant methods	6
2.2.2 Route	6
2.2.2.1 Significant data structures	6
2.2.2.2 Significant methods	6
2.2.3 Waypoint	7
2.2.3.1 Significant methods	7
2.3 Activity Classes	7
2.3.1 AbstractActivity	7
2.3.2 CreateRoute	7
2.3.2.1 Significant methods	7
2.3.3 CreateWaypoint	7
2.3.3.1 Significant methods	7
2.3.4 FinishRoute	8
2.3.4.1 Significant methods	8
2.3.5 GPSService	8
2.3.5.1 Significant methods	8
2.3.6 Home	8
2.3.7 WaypointList	9
2.3.7.1 Significant data structures	9
2.3.7.2 Significant methods	9
2.4 Limitations of the Application	9
2.5 Interaction with Remote Server	9
2.6 Improvements to Android Application	10
2.7 Building and Testing	10
2.7.1 Building	10
2.7.2 Testing	10
2.7.3 Documentation	10
2.7.4 References to external sources	10
3 Website Maintenance Manual	11
3.1 Website Description	11
3.2 Website Structure	11
3.2.1 HTML	11
3.2.2 JavaScript	11
3.2.3 PHP	12
3.2.4 Database	12
3.2.5 Algorithms	12
3.2.6 Files	13
3.2.7 Interfaces	13
3.2.8 Suggestions for Improvements	13
3.2.9 Things to Watch When Making Changes	13

3.2.9.1	Website	13
3.2.9.2	Database	13
3.2.9.3	Physical limitations of the program	14
3.2.9.4	Rebuilding and testing	14
4	Database Maintenance Manual	14
4.1	Description	14
4.2	Structure	14
4.3	Files	14
4.4	Things to Watch When Making Changes	14
5	REFERENCES	15
6	VERSION HISTORY	15

1 INTRODUCTION

1.1 Purpose of This Document

To aid with future maintenance of the project's code base, and to document the idiosyncrasies and intricacies of the code used to develop it.

1.2 Scope

This document should be read by any party that is involved in future maintenance of the application. This document contains details of a number of important factors, including:

Android

- Supported Android SDK versions
- Main data structures
- Complex algorithms
- Suggestions for improvements

Website

- PHP methods used
- JavaScript used
- A basic overview of Google Maps API components used
- Problematic areas of code
- Suggestions for improvements

1.3 Objectives

- To give an overview into the structure of the application
- To aid the package maintainer in spotting problem areas in the project's code base
- To provide the package maintainer a reference for complex areas in the code base

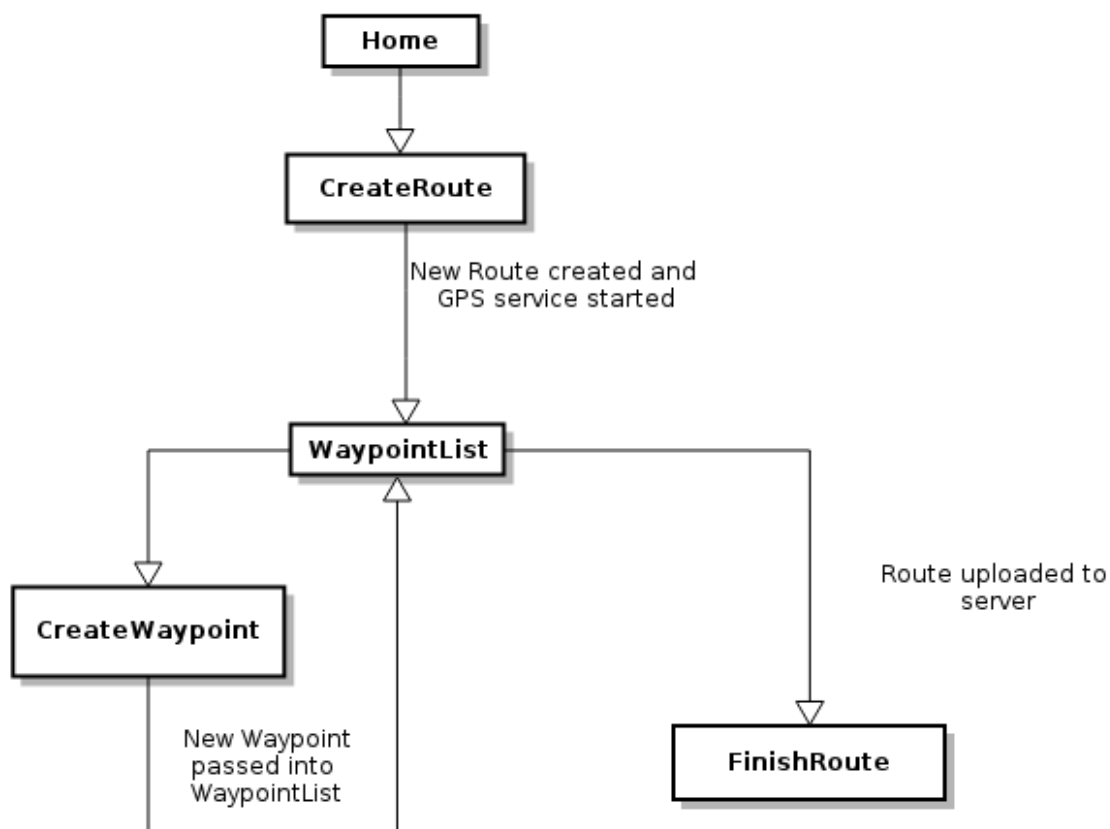
2 Android Maintenance Manual

2.1 Program information

2.1.1 Program Description

The Stumblr Android application is written in Java and is essentially built to help with the creation and upload of walking tours. It allows the user to record a "breadcrumb trail" of GPS coordinates, and gives the ability to add "Points of Interest" (Waypoints) along the way. The user can attach metadata to each Waypoint - including a title, a description, a photo (from the user's gallery or camera), and a GPS location. Before uploading, other metadata is attached to the walk - including time taken and cumulative distance. Once the walk is complete, it is then uploaded (in JSON format) to a remote server, where the data is processed and displayed on a web interface.

2.1.2 Data flow diagram



2.2 Data Classes

This subsection details the significant or complex methods and data structures in each class - large parts of the code base are very well-commented and self-explanatory, so it is only the more complex areas that are mentioned in this document.

2.2.1 StumblrData

2.2.1.1 Significant methods

- **public String sanitiseStringInput**

This method takes a String object and replaces all forbidden characters (more specifically, characters that do not belong to the set "a-z", "A-Z", "0-9", ",.!?:;" and "-". This ensures that String data is safe to transmit to the web server.

2.2.2 Route

The Route class is the main container for the metadata of the walk - including the description, timestamps, the list of coordinates for the breadcrumb trail, and the list of Waypoints (a datatype that will be covered further later in this document).

2.2.2.1 Significant data structures

- **private Stack<Location> coordinates;**

This holds the "breadcrumb trail" - a Stack of Location objects (from the Android API). This is where the Location objects are stored when they are obtained from the GPSService class. Location objects are passed from the GPSService class by using Broadcastsm and then filtering by intent. (This is covered in detail later).

- **private LinkedList<Waypoint> route;**

This holds the list of Waypoint objects - including an instance of a Bitmap image (from the Android API), and Strings relating to the title and description of the Waypoint.

2.2.2.2 Significant methods

- **public float getDistance**

This method calculates the cumulative distance between all of the "breadcrumbs" in the 'coordinates' Stack. The method loops through the list and calls the Android API method **Location.distanceBetween()** on each successive pair of coordinates,. Returns a **float** containing the total distance between each pair of coordinates.

- **public void writeToParcel**

Writes the current contents of the Route object to a **Parcel** object (Android API), to allow the Route to be passed between Activities. Required for implementing the **Parcelable** interface.

- **public void readFromParcel**

Reads a Parcel into the current Route object. Required for implementing the **Parcelable** interface.

2.2.3 Waypoint

2.2.3.1 Significant methods

- **public void writeToParcel**
Writes the current contents of the Waypoint object to a **Parcel** object (Android API), to allow the Route to be passed between Activities. Required for implementing the **Parcelable** interface.
- **public void readFromParcel**
Reads a Parcel into the current Waypoint object. Required for implementing the **Parcelable** interface.

2.3 Activity Classes

2.3.1 AbstractActivity

This class contains all of the common code elements that are used in the other Activities. This includes the code that display the "cancel confirmation" alert dialogue. This class is largely self-explanatory.

2.3.2 CreateRoute

2.3.2.1 Significant methods

- **public void startWaypointListIntent**
This starts the WaypointList Activity after checking that the input fields of the Activity are the correct length.

2.3.3 CreateWaypoint

This Activity is called from the WaypointList activity - and this is where the user can create Waypoints to correspond with Points of Interest (POIs) during the walk.

2.3.3.1 Significant methods

- **public void stumblrOnCreate**
This method is essentially the Activity's constructor - it checks the Bundle data passed into the Activity - checking for a Location object or a Waypoint object. The object is then loaded into the appropriate instance variables (depending on the type). The last thing this method does is use the Android API to check if certain capabilities are enabled; namely, a camera and gallery.
- **public void finishWaypoint**
This just validates the data entered on-screen, warning the user with a **Toast** if any data entered is invalid. A timestamp is then applied to the current Waypoint, and the Waypoint object is put into a Bundle as a return value for the Activity - this Bundle is returned to WaypointList (as it is instantiated with **startActivityForResult()**).
- **public void getImage**
Gets an image from user's camera or gallery, after checking which is available. An Activity is dispatched (based upon what is available) using **startActivityForResult()**.
- **protected void onActivityResult**
This method handles return values from the camera or gallery Activities dispatched in the previous method. Depending on the activity that returned the image, it will be resized and cast to a **Bitmap** for easy conversion to Base64, and submission (via JSON) to the web server.

2.3.4 FinishRoute

This Activity is responsible for converting the (completed) Route object to JSON and uploading it to the web server, ready for displaying on the web interface. As the data structures dealt with in this class are generally simple (and have already been covered), they are not mentioned in this section.

2.3.4.1 Significant methods

- **public void stumblrOnCreate**
This method is fairly simple - it unbundles the Route object from the extras passed in from the calling Intent, and puts it into memory. The remaining Route metadata is then calculated and displayed onscreen.
- **public void postData**
This method simply uses the Android API to check if the device has an internet connection - if so, the NetworkTast is executed, and the data is sent.
- **public JSONObject getData**
This converts the Route object into JSON ready for transmission to the web server. This method uses the **JSONObject (org.json.JSONObject)** class from the standard Java JSON libraries.
- **public String encodeToBase64**
This is fairly simple - a **ByteArrayOutputStream** decodes a **Bitmap** object into raw data - which is then encoded using Base64

2.3.5 GPSService

GPSService uses a Foreground Notification (to avoid getting killed by Android) and implements **Location-Listener** so that it is able to subscribe to Location update (provided by GPS). The **Location** objects are then sent back to the main Activity by using a Broadcast.

2.3.5.1 Significant methods

- **public int onStartCommand**
This is a rather straightforward method, but it looks a little complicated. It subscribes the current class to receive **Location** updates and places a persistent **Notification** in the user's Notification Centre. **Be aware: There is officially deprecated code inside this method so that the persistent Notification can be shown on older versions of Android.**
- **public void onDestroy**
This method tells the class to stop receiving Location updates when killed - this ensures that the device's battery is not wasted.
- **public void onLocationChanged**
This method is very simple, but its function may not be immediately obvious. It is called when a **Location** update is received by the **GPSService** class, and it bundles the received **Location** object and sends it as a **Broadcast** to the system, where the message is picked up by **WaypointList** and added to the current Waypoint.

2.3.6 Home

The Home class contains no complex methods as it is approximately 35 lines in length and only contains code for one button!

2.3.7 WaypointList

This is the main Activity of the application in that most of the user's time is spent using it. From this Activity, the user can view, add to, and edit the current list of Waypoints. The user can also decide to complete the walk, dispatching the FinishRoute Activity.

2.3.7.1 Significant data structures

- **private ListView listView**

This ListView object is used to display and manipulate the Route's Waypoints on-screen. The user can view a list of the current Waypoints, or edit them by tapping each entry in the list.

2.3.7.2 Significant methods

- **protected void onActivityResult**

This method is called when a dispatched Activity (in this case, CreateWaypoint) exits and returns with a valid Waypoint object - the new Waypoint is added to the list and the ListView is refreshed.

- **private void startGPSService**

There is a private class declaration within this method that contains methods to add coordinates received from the GPSService to the current trail of **Locations**. If GPS is disabled, the method prompts the user to enable it.

- **public void onDestroy**

Called before the Activity is destroyed (part of the Android application lifecycle). This sends a signal to the running GPS service to halt and exit (thus removing the **Notification** at the top of the user's screen).

2.4 Limitations of the Application

There are a number of limitations to the application in question, the principal of which are the Android API versions that it supports. Currently, the application supports (and has been fully tested on) API/SDK versions **8** to **22** - which correlates to Android versions **2.2** through **4.4**.

There are also a number of important considerations when it comes to battery and memory usage. Each Waypoint (including thumbnail image) is stored entirely in working memory (RAM) - theoretically, this limits the number of Waypoints that a low-RAM device can hold; although in practice, the amount of data in memory (even on large walks) is relatively small, and memory usage is a very minor consideration. Certainly, we experienced no issues on an older Android **2.3** device with severe memory limitations, with walks of 20 Waypoints and above.

The application is also rather battery intensive on longer walks; the GPS service, while being useful for regularly fetching location data - uses an enormous amount of battery power, particularly if the GPS signal is bad (due to overcast weather, large buildings in the surrounding area, etc). This is sufficient to flatten some more power-hungry devices with 5-6 hours' usage (tested overnight on a tablet device).

2.5 Interaction with Remote Server

The interaction with the server is kept as simple as possible. The Java objects created within the application are converted to JSON (a human-readable format useful when transmitting complex objects), and transferred to the server via HTTP POST. The HTTP interaction is completed with the Apache Commons HTTP libraries, and the code used to interact with the server is currently in the FinishRoute class.

2.6 Improvements to Android Application

There are several existing bugs and shortcomings within the application that were discovered after release; these would make a good starting point for any future maintenance. These were discovered through our extensive testing processes, and represent the sum total of the bugs we have found to date.

- Multiple images on each waypoint:
Due to a misinterpretation of the customer's requirements, the application does not support inclusion of multiple images per Waypoint for each Walk.
- Editing of Route title:
Discovered during acceptance testing - another result of misreading requirements. The Route title cannot be created once the Route has been created.
- Refreshing of ListView in onRestoreInstanceState method:
This was discovered during our final pre-release testing - when the user has a list of Waypoints on the WaypointList screen, minimises the Activity and the Android system kills the sleeping Activity, the data is restored to the application, but the Waypoints don't appear on the screen, due to the ListView not being updated. The application data is restored, it just doesn't appear until another Waypoint is added.

2.7 Building and Testing

This subsection details the tools and methods that were used to create and compile the application and documentation.

2.7.1 Building

The build tools used in the creation of the Java code base were Android Studio, coupled with the Gradle build system. The supplementary files inside the Git repository are mostly to ensure proper functioning of Gradle and the Android Studio IDE.

2.7.2 Testing

The testing code for this assignment was written using JUnit - a testing suite that complements the Java language.

2.7.3 Documentation

All documentation for this project (where possible) has been written and rendered using L^AT_EX. L^AT_EX can be a challenge to write for new users, so a procedure was defined for team members who struggle getting it to work; writing documents in Markdown and then converting them. Markdown is a minimal, intuitive markup language (also used on GitHub for formatting text files) that aims to be easy to write. This was excellent for our needs, as Markdown syntax can be easily converted to L^AT_EX syntax using a system like Pandoc.

Markdown: <https://daringfireball.net/projects/markdown/>

Pandoc: <http://johnmacfarlane.net/pandoc/>

2.7.4 References to external sources

References to external sources are usually contained inside inline comments in the Java code; *for example*:

```
// See:  somewebsite.com/url/resource.html
```


3 Website Maintenance Manual

3.1 Website Description

This website is a walking tour viewer, it pulls data from a database which is provided by the android application Stumblr. It then uses this data in conjunction with the Google maps API on the website to display the tour on the map, including information on the locations visited on the route. It also provides a gallery bar of all the images of the tour selected if there are any.

The system uses a database to store details on the tours captured using the Android app. The details stored in the database are then displayed on the website using a combination of SQL commands and PHP scripts. The database itself contains three tables, all storing different information on the tours. The tables store information on each walk, their location and also the path for each walk.

3.2 Website Structure

I will show the structure of this website in two ways; first I will provide you with the flow of the website to show you how it works then, I will go through the code saying what the methods and sections of the code do.

There are four main components to the websites; HTML, JavaScript, PHP and the database. All of these all work together to create the web page Stumblr.

3.2.1 HTML

The HTML is what I have used for the design such as having the body of the web page centred using a div called divwrapper which has been set to be 80% width of the browser screen. The HTML includes the divs tour, map and bar. The tour div is where information of the tour is presented when the tour is loaded this will contain the name and description of the tour plus any other information held in the database. The map div is used to hold the Google map which is created by the JavaScript this is set to the width of the body. The bar div is where the gallery of images is shown as a scrollable list of images with their title and description. The HTML also has the form which is used to select and load a tour which it does by sending the ID of the tour back to the page using POST. It gets the ID and title of the tour in this from by using php to get the information from the database.

3.2.2 JavaScript

The JavaScript is used to create the map and gather information from the database to populate that map. The JavaScript in this web page has 3 functions `downloadUrl()`, `bindfInfoWindow()` and `load()` which I will talk about now :

`downloadUrl()` - this function is just a basic download function to download a url, in this case we are using it to download an XML file. The XML file contains all the details of the selected tour.

`load()` - this function loads up the google map, by default it displays nothing as no tour is selected. If a tour is selected it will display on the map. First it creates a variable called `map` which is a new Google map, the next few variables are first where the map should be centred which is the co-ordinates of the centre of Aberystwyth town, next is the zoom level of the map which controls the level of zoom on the co-ordinates and the last one is the map type which is roadmap so people can see the roads and also use Google street-view. Next the `downloadUrl()` function is called to download `XMLcreator.php`. This contains the details of the tour. Next the variable `markers` are created by getting information from the XML document. A loop is used to set the variables for information to be displayed as a marker, below is the code for a marker:

```
var marker = new google.maps.Marker({map: map, position: point,
icon: "images/mapiconscollection-numbers-fa0505-classic/number_" + i + ".png"});
bindInfoWindow(marker, map, infoWindow, html);
```

Then info window is also added here using the information from the tour as you can above.

The next part is where the function downloads the `xmlpath.php` file that is used to create a route between the markers on the map. This works a little like the inner function that creates the markers and info windows

as it uses a variable paths instead of markers but this function also uses a variable pathco which is an array to hold all the co-ordinates of the path to be used later. The loop gets the length of the route from the XML file, it then gets the co-ordinates of the route taken and sets them to a variable called point. This variable is then pushed to the pathco variable using the following code: `pathco.push(point);` this is used to push all the variables to this array to create an array of the path co-ordinates. Below this you will see the path options being set such as the path variable being set to the pathco array, and other options such as `strokeColor`, `opacity` and `weight` which can be altered to create a different route look, then below the path is set to the map.

`bindInfoWindow()` - This function is used to tie all the markers and infoboxes to the map and where a listener is added that when markers have been clicked they open an infobox with the information in the html variable.

3.2.3 PHP

The PHP used in this website is mostly used to get information from the database but there are a few lines that are used for other things.

At the top of the page you will see: `session_save_path(); session_start();`

This is used to set where sessions will be saved to a tmp folder as most web servers will save sessions to a tmp folder if the path is an empty string. Then it starts off the session using the `session_start()`.

Then the variable `$walkID` is created and set to zero so when first opened the web page will come up with a default map instead of a tour. Then if somebody loads up a tour it will send a POST back to the page which is why the next line is the variable being set to `$_POST['tours']` so `$walkID` will be equal to the tour that has been selected. Then a session variable is created which is also equal to the POST which is used in the XML files to get the correct tour for the Google map.

Then the next few lines are used to connect to the database and then query tables in that database to get information which is to be used on the webpage. Next there is an IF statement just encase there are problems connecting to the server, including the error message "Failed to connect to MySQL: ". PHP is next used to create a drop down box that contains the names of all the tours on the database, it uses a query that is created at the top of the page to get the information from the database. Then the information about the tour that is loaded is displayed above the map. This includes the tour's title, long description, short description, time and distance. Finally there is the gallery where we use the same query used above to display the images for each tour. The image is base64 encoded and must be decoded before being displayed. It is displayed along with a title and description.

3.2.4 Database

The first table is used to store the walks, it holds information on the title stored as a varchar, short description stored as a varchar, long description stored as a varchar, time taken stored as a float and distance also as a float. The primary key is used to reference the other details of each tour in the other tables. The next table used contains details on each location for a walk. The table stores each location's title, short description, long description all as a varchar, longitude, latitude and a time stamp as a float and finally an image stored as long text due to the images being base 64 encoded. The table also has an ID as a primary key, as well as a foreign key which links the location to a walk in the walks table. The final table used in the database stores a path for each walk. Therefore it has a foreign key integer which references to the walk it represents, a value for the longitude and latitude which are both stored as floats.

3.2.5 Algorithms

The main algorithm used for this webpage is creating an XML file of information from the database. This is used in both `XMLcreator.php` and `xmlpath.php`. They do this by connecting to the database, and then pulling data from the table that the information is in. Next it iterates through the rows adding XML nodes for each part, which is then used in `index.php` when the function `downloadUrl()` is called to get this information which it loops through to get all of the information.

3.2.6 Files

For the website there is an extra file the Cascading Style Sheet, which is used for the design of the page such as the background and how big the body of the website is going to be and also all the sizes of the divs.

In order to access the data in the database three files containing PHP scripts were used. Two of the files were used for converting the data into XML, one for the locations on a tour and another for the path. Then the XML could be used to display the tours in conjunction with Google Maps. The way these scripts work is to first connect to the database, then they use SQL queries to find which data in the database they need depending on which tour is selected. Finally they pull the data from the database and insert it into a new XML file. The final file which is used is another PHP script and this decodes the incoming data in the form of a JSON file and inserts them into the database. Firstly the script establishes a connection to the database, then it takes in the JSON file and decodes it using a PHP function called `json_decode`. Next it stores the decoded data as variables and these variables are using along with SQL commands to add the data to the database.

3.2.7 Interfaces

The main interface with the website is the Google map which I have gone over in some length in the website structure above. If you wanted to know more about this interface then there is a Google Maps API. The only other interface with this website is the drop down box and load tour button which is generated with some PHP and they are located in a form which sends the id of the tour back to the page so it loads that tour up from the database.

3.2.8 Suggestions for Improvements

A few improvements I could suggest would be to improve the look of the gallery bar at the bottom of the page so the titles, pictures and description line up properly, this probably be done with some new divs in the CSS file. Another improvement I could suggest is the way information about the tour is loaded above the map which could be done with several little things such as a border or the changes to the div and how the information it presented with the `<p>` tags. Another suggestion would be the background of the web-page as I have seen it tile at the bottom when there is too much information on the page that it makes it tile this could probably be fixed by changing the CSS for the background or you could get a bigger picture. Another suggestion for improvement would be to add PHP unit tests to website as they are none right now, it would be good to have these so if something goes wrong you will know where the problem is.

3.2.9 Things to Watch When Making Changes

There are many things to watch for when making changes I will now go through some with the website and the database.

3.2.9.1 Website The thing you most want to watch out for when making changes to the website is the JavaScript as one small change and a whole bunch of problems can happen. I would recommend having a backup of the file working before making changes to see where you might have gone wrong. Another way you could prevent from getting errors with the Google maps JavaScript is to go through their website as they have many examples of a working Google map. Also another thing to watch out for when making changes to the website is that you use the right queries for getting information out of the database. Also when testing the website if you would use the browsers debugger like Firefox's this can tell you where you are going wrong.

3.2.9.2 Database Changing the overall structure of the database will have knock on effects on the rest of the system. Most of the PHP code used for accessing the database uses variables and statements which rely on the names of columns in the database. The areas that it would affect would include pulling data, decoding the JSON and inserting data into the database. Therefore it is important that if any changes are made to the database they are made to every PHP file.

3.2.9.3 Physical limitations of the program A physical limitations of our website is that the database is held on a different database to the website which means if it goes down there is now way of using the information on the database with the website. This means if it went down the website would not be able to get any information for the tour leaving it mostly blank and useless.

3.2.9.4 Rebuilding and testing There is no need for rebuilding of the website as PHP is an interpreted language so it does not need to be rebuilt. For testing, there are no PHP unit tests in the website - so you can only browser-test it to ensure that it works in the most popular browsers.

4 Database Maintenance Manual

4.1 Description

The system use a database to store details on the tours captured using the android app. The details stored in the database are then displayed on the website using a combination of SQL commands and PHP scripts. The database itself contains three tables, all storing different information on the tours. The tables store information on each walk, their location and also the path for each walk.

4.2 Structure

The first table is used to store the walks, it holds information on the title stored as a varchar, short description stored as a varchar, long description stored as a varchar, time taken stored as a float and distance also as a float. The primary key is used to reference the other details of each tour in the other tables.

The next table used contains details on each location for a walk. The table stores each locations title, short description, long description all as a varchar, longitude, latitude and a time stamp as a float and finally an image stored as long text due to the images being base 64 encoded.. The table also has an ID as a primary key, as well as a foreign key which links the location to a walk in the walks table.

The final table used in the database stores a path for each walk. Therefore it has a foreign key integer which references to the walk it represents, a value for the longitude and latitude which are both stored as floats.

4.3 Files

In order to access the data in the database three files containing PHP scripts were used. Two of the files were used for converting the data into XML; one for the locations on a tour, and another for the path. Then the XML could be used to display the tours in conjunction with Google Maps. The way these scripts work is to first connect to the database, then they use SQL queries to find which data in the database they need depending on which tour is selected. Finally they pull the data from the database and insert it into a new XML file.

The final file which is used is another PHP script and this decodes the incoming data in the form of a JSON file and inserts them into the database. Firstly the script establishes a connection to the database, then it takes in the JSON file and decodes it using a PHP function called `json_decode`. Next it stores the decoded data as variables and these variables are using along with SQL commands to add the data to the database.

4.4 Things to Watch When Making Changes

Changing the overall structure of the database will have knock on effects on the rest of the system. Most of the PHP code used for accessing the database uses variables and statements which rely on the names of columns in the database. The areas that it would affect would include pulling data, decoding the JSON and inserting data into the database. Therefore it is important that if any changes are made to the database they are made to every PHP file.

5 REFERENCES

- [1] PAVLIC, T. Programming Assignment template for L^AT_EX.
<http://www.latextemplates.com/template/programming-coding-assignment>.
Accessed: Oct 23, 2013.

6 VERSION HISTORY

Author	Date	Version	Change made
CCN	07/11/2013	1.0	Updated order of authors on cover

Group 10 Test Log Form

February 16, 2014

SE.10.TL.1

Version: 1.0
Status: Draft

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2013

Contents

1	TEST LOG FORM	3
1.1	Tests	3
2	REFERENCES	8
3	VERSION HISTORY	8

1 TEST LOG FORM

1.1 Tests

Test ID	Pass / Fail	Additional notes	CCF / issue num- ber	Image ref
1	Pass			Fig1
2	Pass			Fig2
3	Pass			Fig3
4	Pass			Fig4
5	Pass			
6	Pass			
7	Pass			
8	Pass			
9	Pass			
10	Pass			Fig5
11	Pass			
12	Pass			Fig5
13	Pass			
14	Pass			Fig5
15	Pass			Fig5
16	Pass			Fig5
17	Pass			Fig5
18	Pass			Fig5
19	Pass			
20	Pass			Fig6
21	Pass			Fig7
22	Pass			Fig6
23	Pass			
24	Pass			
25	Pass			Fig8
26	Pass			
27	Pass			
28	Pass			
29	Pass	The waypoint screen must be refreshed in order to see its contents.	(Android Repo) #6	
30	Pass			
31	Pass			
32	Pass			
33	Pass			
34	Pass			
35	Pass			
36	Pass			
37	Pass			
38	Pass			
39	Pass			
40	Pass			

Fig1

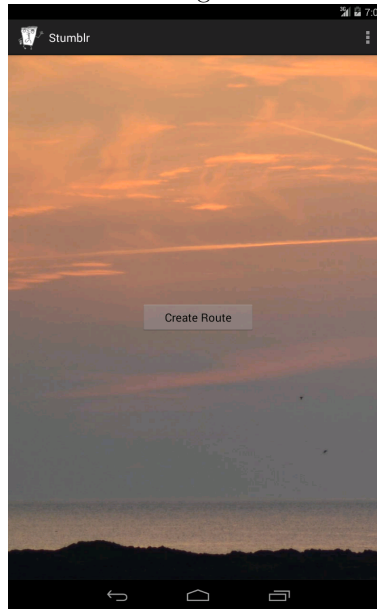


Fig2

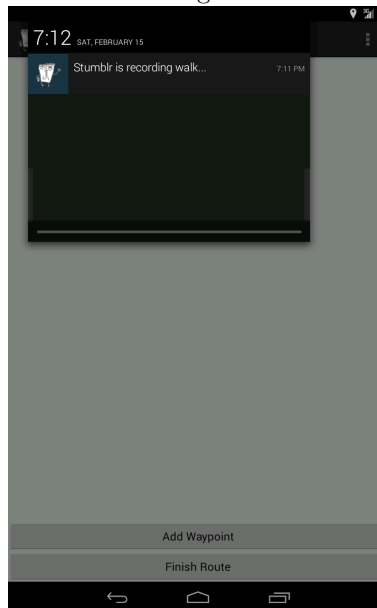


Fig3

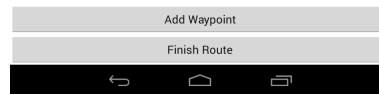


Fig4

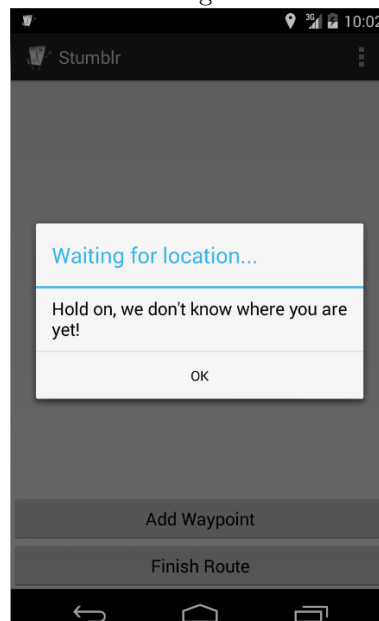





Fig5

Tour Submit - Mozilla Firefox

stumblr/DOCS/Controlled/... Tour Submit

users.aber.ac.uk/mal60/group_project/submit.php

17	6	Jlfdjd	52.4167	-4.0652	Khfgdc	1391180000000	
18	8	Location	52.4142	-4.08362	Description	1392310000000	
19	9	Test Location	52.4138	-4.08314	Test	1392310000000	
20	10	Garden	52.4142	-4.08386	Heres a test picture	1392310000000	
21	10	Another Picture	52.4143	-4.084	Heres another test picture	1392310000000	

Path

ID	WalkID	Latitude	Longitude
1	1	52.4162	-4.06606
2	1	52.4162	-4.06625

Fig6

Stumblr

Route Title

123

Short Route Description

Long Route Description

Next

The title is too short. It must be > 3 characters.

Fig7

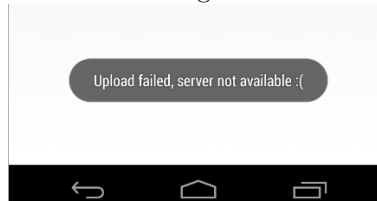
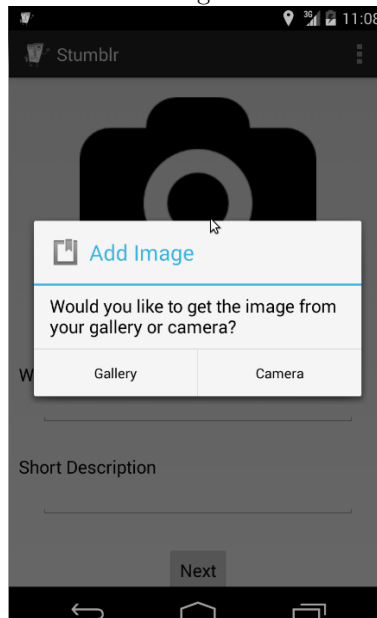


Fig8



2 REFERENCES

3 VERSION HISTORY

Author	Date	Version	Change made
CCN	07/11/2013	1.0	Created template

Group 10 Test Specification

February 17, 2014

SE.10.D2

Version: 1.4

Status: Release

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2013

Contents

1	INTRODUCTION	3
1.1	Purpose of This Document	3
1.2	Scope	3
1.3	Test Plan	3
1.4	Introduction to the Test Procedure	3
1.5	Deliberate Test Data Ambiguity	3
2	TEST TABLE	4
2.1	Tests	4
3	REFERENCES	9
4	VERSION HISTORY	9

1 INTRODUCTION

1.1 Purpose of This Document

The purpose of this document is to provide a plan for the implementation of thorough tests on all required functionalities, as per the customers specifications. This document will show we have managed to expand on each functional requirement to make a suitable list of conditions, to test how well our software meets each requirement.

1.2 Scope

This document should take into account the required functionality of the project. This document includes; a Test Plan, outlining the ways in which we intend to test our system, and a Test Specification, which represents the set of System Tests we intend to use to outline how our software meets the set requirements.

1.3 Test Plan

We intend to test our system in two main ways to ensure the production of reliable software. These are:

- A set of Unit Tests to be written and run by the programmers of the project. These will ensure that, no gaps in functionality arise.
- A set of Black Box System Tests to reliably compare our final product with the product that the customer has specified.

1.4 Introduction to the Test Procedure

The purpose of this test specification is to specify in detail each of the system tests that will be executed to validate our conformance to each of the functional requirements as laid out by the customer. The table below will list a reference to functional requirement to be tested, and should be reproducible with the exact inputs and outputs listed. Each test will have:

- A unique test reference.
- A listing of the functional requirement(s) being tested.
- A description of the test.
- The exact input for the test.
- The expected outcome for the test.
- The expected outcome for the test.
- The criteria for whether that test has passed or failed.

1.5 Deliberate Test Data Ambiguity

Some of the test data described in the tables below is ambiguous. This is on purpose - we have not designed and implemented the system to a level that would justify designing specific test data sets; we will design more appropriate test data sets (along with unit tests) when the system is implemented more fully.

2 TEST TABLE

2.1 Tests

Test	Req	Context	Input	Output	Pass Criteria
1	FR 1	On startup the user will be given the choice to create a new walking tour.	Start app.	"Create new walking tour" will be displayed.	App should start up correctly and the correct UI should be displayed.
2	FR 1	When tour is created, user should be prompted for basic tour details.	Select create new walking tour.	Details regarding tour and GPS recording should start.	App will prompt for details and GPS device will be enabled.
3	FR 1	After recording starts, options for cancelling recording, saving the tour to server, adding locations (inc text and photos) should be displayed.	Start recording.	Options should be displayed.	Options for cancelling recording, saving the tour to server, adding locations (inc text and photos) are displayed.
4	FR 1	GPS connectivity lost after recording has started.	Turn off GPS and location services during route recording.	Error message displayed stating connection is lost, will maintain session untill connection is reestablished.	Session with remain at last known location.
5	FR 2	User enter a single word name for route, e.g "Aberystwyth". Name will appear on database.	Enter "Aberystwyth" as route name.	Route title is Aberystwyth and is displayed on database.	Route title matches input and is displayed on database.
6	FR 2	Enter short description of walk/route with up to 100 characters.	Enter 100 characters for short description.	Description will be added and entered into database.	No error messages displayed and description appears on database.
7	FR 2	Enter short description of walk/route with more than 100 characters.	Enter 101 characters for short description.	No new characters can be added to the text box when it is full.	Any character that is added past the limit will not be added to the data field.
8	FR 2	Enter long description of walk/route up to 1000 characters.	Enter 1000 characters for long description.	Description will be added and entered into database.	No error messages displayed and description appears on database.
9	FR 2	Enter long description of walk/route with more than 1000 characters.	Enter 1001 characters for short description.	No new characters can be added to the text box when it is full.	Any character that is added past the limit will not be added to the data field.

10	FR 3	Add location to the tour with map coordinates (latitude and longitude).	While route is recording, select "add location".	Location should be added to the database with latitude and longitude.	Location should be added to the database with latitude and longitude.
11	FR 3	Add name for location.	When add location has been selected, add a name; e.g "Test location".	"Test location" should be set to the location's title at coordinates.	"Test location" should be set to the location's title at coordinates and stored correctly at set location in database.
12	FR 3	Add description for location.	Add description, e.g popular beverages and event nights.	Add valid location description, e.g. "Magic Mondays".	Location description added to location on database.
13	FR 3	Add a timestamp for location.	User selects "add location".	Timestamp automatically added.	Timestamp should be added to location showing when it was added onto the database.
14	FR 4	Adding photos to the walks.	User should be able to add one or more photos, captured from camera or from the device's photo library. User selects add photo to location while route is recording.	Photo is added to location user is then requested to enter name and description for location.	Photo added to database. Photo is set to location and timestamp added. Prompt to enter name and description for location is then displayed.
15	FR 4	After photo is taken, give location of photo a name.	Add name of location to photo. e.g "Harry's Bar".	Location for photo set to "Harry's Bar"	Photo location name in database set to name entered by user.
16	FR 4	After photo is taken and input is prompted, add description of location in photo.	Add description, e.g "Most popular pub on route".	Input criteria e.g "Most popular pub on route" added to photo location.	Description added to photo location in database.
17	FR 5	Cancelling walk - stop recording and don't save.	Without any added locations, select stop recording button.	Route is stopped and deleted.	No information stored on database. User returned to main menu.
18	FR 5	Cancelling walk - stop recording and don't save where locations of interest have been added.	Select stop recording and don't save.	Route is stopped. Warning displayed that if the route is cancelled all saved locations will be deleted. Proceed.	All information stored on this route is removed from the database.

19	FR 6	Test that tour has been sent to the server correctly.	Standard data you would expect to see; Location name, title, long and short descriptions, a list of GPS coordinates for the walk from start to end, with a time stamp for a location, list of locations with associated information, photos with associated information.	Message to confirm that the data has been sent.	A new tour should now be saved in the database containing all the entered details.
20	FR6	Test user has entered data into the title and short description fields before continuing to the next screen.	Attempt to continue to next screen without entering data.	Error message to instruct the user to enter data into the title and short description fields.	Error message should appear and nothing new should be saved in the database.
21	FR6	Test sending the tour with no internet access. (WiFi/Mobile)	Data for a standard tour.	Error message should warn the user that the tour can not be submitted at this point.	Error message should appear and no data sent. The current tour should be cached so it can be sent at a later time.
22	FR6	Test sending to server when internet drops out.	Tour data sent and internet connection disabled.	Error message displayed.	Error message display and tour data cached.
23	FR6	Test that the app continues to upload the route when it has lost focus.	Create a normal route and when "Upload Walk" is pressed immediately press the "Home" icon.	The app will have lost focus.	When the app is resumed the upload confirmation message will be shown.
24	FR6	Test for data integrity checking on the server.	Send a route that doesn't have a matching md5-hash code.	Server returns a code that is not a "200" code.	No new route is added to the database.
25	FR6	Test that the app can gain access to the camera app.	When adding an image to a waypoint click "Gallery" and choose a stored image to load.	The app returns to the add waypoint screen.	The chosen image is displayed in the waypoint screen.

26	FR 7	Test to check that the app can save a full tour when the app is shut down at the Unfinished Route screen.	Add data about the tour up until the screen for submitting to the server, then close the app.	App should close without delay.	When the app is re-opened it should have saved details about a full that is ready to be submitted to the server.
27	FR7	Test to check the app saves data at the Create Route screen.	Add title, short and long description about a tour then close the app.	No output.	When app is re-opened it will display the details entered about the tour.
28	FR7	Test to check that the app saves data when closed at the Add Waypoint screen.	Waypoints added to tour and app closed.	No output.	When the app is re-opened it will display the incomplete tour with the waypoints.
29	FR7	Test to check that the app saves data when closed at the Waypoint list screen.	Details about a Waypoint added and app closed.	No output.	When the app is re-opened it will display the details about the waypoints.
30	FR7	Test to check that when the app is closed a partial walk can be saved.	Add data about just one location then close the app.	No output	When the app is re-opened the user should be able to continue the tour from where they quit.
31	FR 8	Test viewing a saved tour's location on the Walking Tour Display.	Selected tour from the saved tours list.	The WTD should display the current tour on the map.	The tour should display with the right co-ordinations and also with flags/markers to click on for images and descriptions.
32	FR8	Test viewing details of locations for a tour.	Selecting a tour from the saved list, then clicking a location in that tour to reveal it's details.	When a location is clicked a pop-up should appear with its details and image.	The locations details and image are correctly displayed in the WTD.
33	FR8	Test switching between saved tours.	Switch from one tour to the other on the WTD.	The new tour should be displayed on the map.	Waypoints and location change on the map.
34	FR8	Test switching between waypoints.	Click between one waypoint to another.	The information for the new location should be displayed.	The information on the old location should have disappeared and the new locations information should be displayed.

35	FR8	Test opening a way-point without an image.	Click on a location that doesn't have an image.	The information of the location is displayed.	The location's information is displayed without an image.
36	FR8	Test opening a way-point with an image.	Click on a location that has an associated image.	The information of the location is displayed.	The location's information is displayed and contains an image.
37	FR8	Try to view a route that is not in the database.	Check that only the routes that are contained in the database are displayed in the "Tour Selector" drop down box.	No output.	No "Phantom Routes" can be seen in the drop down box.
38	FR9	Test to confirm that the data has been successfully saved on the server.	Details about a tour sent to the server.	A confirmation message is displayed.	The tour should now appear as a saved tour in the database.
39	FR9	Test that an error is caught when data cannot be saved on the server.	Details about a tour sent whilst the server isn't working.	An error message is displayed.	An error message should appear telling the user that the tour cannot be sent and the tour should be cached.
40	FR9	Test for data sanitisation on the server.	Send a route that contains invalid characters in one of its data fields.	No output.	The bad data's corresponding entry in the database will not contain the invalid characters.

3 REFERENCES

- [1] PAVLIC, T. Programming Assignment template for L^AT_EX.
<http://www.latextemplates.com/template/programming-coding-assignment>.
Accessed: Oct 23, 2013.

4 VERSION HISTORY

Author	Date	Version	Change made	CCF
CCN	07/11/2013	1.0	Created template	n/a
CCN	15/11/2013	1.1	Added test table	n/a
CCN	15/11/2013	1.2	Update headers	n/a
DEC	15/11/2013	1.3	Spell checked	n/a
ATI	1/02/14	1.4	Updated tests to reflect current spec	#38

Group 10 Project Plan

February 17, 2014

SE.10.D1

Version: 1.2

Status: Release

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2013

Contents

1 INTRODUCTION

1.1 Purpose of This Document

The purpose of this document is to show that we have met the outlined objectives specified by the client.

The main objective is to create a system that enables users to collect and compile data about a walking tour - through the use of a smartphone application and a web interface. The smartphone application will be able to use the device's GPS functionality and internet connectivity to collect and upload the data, and the web interface will display the data in a sensible fashion. All information will be stored in a database, that will allow insertion of data by the phone application, and retrieval by the web interface. The data should include a photo, textual descriptions, and titles, as well as GPS location data and other metadata including timestamps, and so on.

This document will show we have managed to simplify these into a set of manageable goals. Gantt charts and GitHub's contribution analysis functionality will be used in conjunction to show and monitor the progress of the project's major tasks and other milestones. We are also implementing a risk analysis system that should alert us to the various problems that we may face whilst completing our objectives, and also allow us to mitigate the risks involved.

1.2 Scope

This document should take into account the specifications of the project. This document includes an overview of our proposed systems - which will encompass our choice of platforms, some high-level architectures and a description of prospective users. The document also contains; a use case diagram giving an overview of how the app and the web systems will interact, mock-ups and descriptions of the UI design and how both applications will interact with the user, a Gantt chart which displays the start and end dates for the main milestones, and a risk analysis for the project.

1.3 Objectives

These main objectives are to show our initial plans for the project. These goals include an overview of our proposed system, a set of use case diagrams, user interface designs, a Gantt chart, and a risk analysis.

- Produce an overview of our proposed system, which matches our client's specifications.
- Produce a set of detailed use case diagrams to define the interactions with the system components and users and provide them with a concise explanation, along with example usage scenarios.
- Create a UI for the Android and web systems we will employ, with a succinct explanation for our chosen design.
- Employ the usage of a Gantt chart, to document our expected progress as a team.
- Highlight aspects of our project plan which may cause us difficulties in completing our assigned work in a timely manner.

2 SYSTEM OVERVIEW

2.1 Introduction

Our proposed system is an application that will allow the user to create guided tour, with waypoints that include photographs and descriptions. We will implement the ability to record new routes via an Android interface and view previously recorded routes via a web interface.

2.2 High Level Architecture Diagram

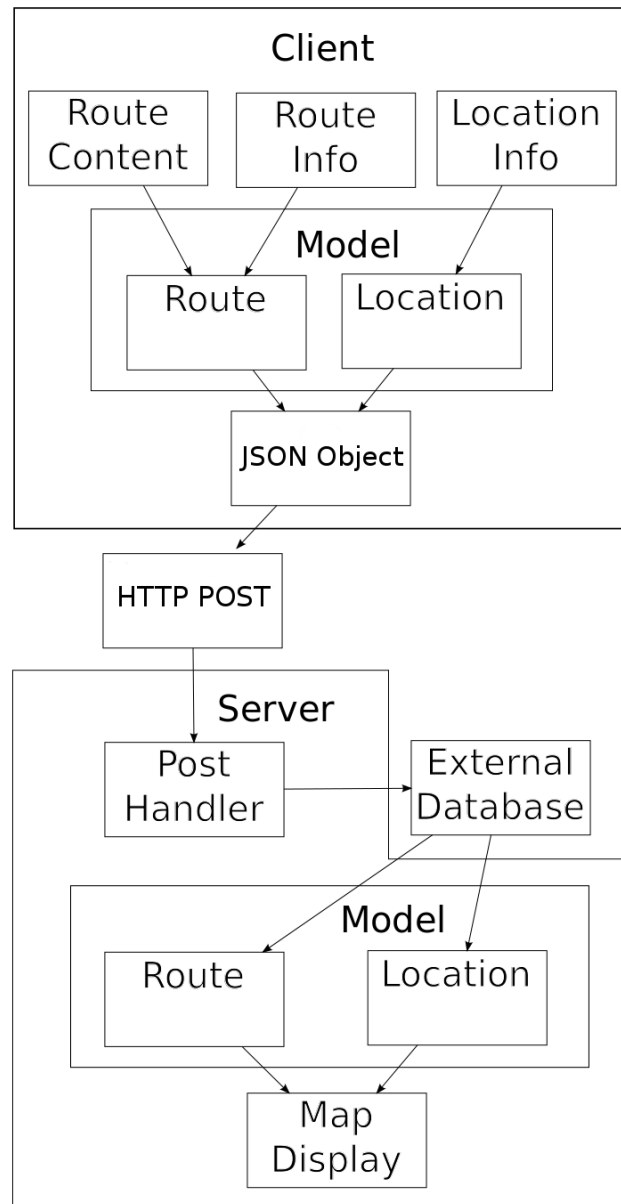


Figure 1: A diagram showing the high-level architecture of the system.

2.3 Client and Server Languages

The server will use a MySQL database as backend and a PHP, HTML, and JavaScript frontend. The Android application will use Java and the associated Google APIs as programming languages.

The website will be hosted on <http://users.aber.ac.uk/>, due to a number of factors that will be detailed in the design document.

3 PLATFORMS AND HIGH LEVEL ARCHITECTURE

3.1 Mobile Architecture

3.1.1 Operating System

It is stated in the specification set by the client that the system is to be designed for Android mobile phones. We will be targeting Android SDK/API version 8 and upwards - this translates to Android 2.3 (Gingerbread) and higher.

3.1.2 Location Derivation

The user's location will be derived with GPS using the Android API and the smartphone's GPS connectivity.

3.1.3 Connectivity

The application will need to use several aspects of a mobile phone's connectivity - for a start, when the user is creating a walk, each waypoint will need to be marked with GPS coordinates. Then a further functional requirement specifies that the application must be able to upload each route in an HTTP POST to a remote server. This obviously requires the utilisation of internet connectivity - the application will support uploading over both WiFi (or 802.11 Wireless LAN) and over the mobile network.

Being able to upload over WiFi and the mobile network introduces problems with the upload of the "route package" - uploading over the mobile network will take considerably longer than over WiFi, not to mention increasing the impact on the user's battery life. Because of this the system will be designed to alter the size and quality of each image within the route package. The images will be "shrunk" in size and strong JPEG compression will be used to reduce file size where possible.

The problem of connection instability must also be addressed. The application must be robust enough to reliably deliver a finished route over HTTP - a problem when using a mobile network or WiFi. The application will be able to handle connection outages and instability, and subsequently resume or retry transmission when a stable connection is restored.

3.1.4 Usage of Other Hardware

The application must be able to attach photos for each waypoint along the route. These photos will be accessible either through the device's camera, or through the device's gallery. This means that some Android API calls must be used to be able to access the camera functionality and the device's file system.

3.2 Web Architecture

3.2.1 Server-Side Scripting

PHP is the chosen server-side scripting language. It is sufficiently fast and light enough to serve our needs, and has a reputation for stability. PHP is also excellent for HTML page generation, MySQL database integration, and dynamic content display.

3.2.2 Client-Side Scripting

For more dynamic areas of the web interface, and client-side access to some of the APIs provided by Google, we will be using Javascript.

3.2.3 Database

MySQL is a popular and widely used database platform which will be used to store the data from each route. This will be used as a backend for both the Android application's upload function and the web interface.

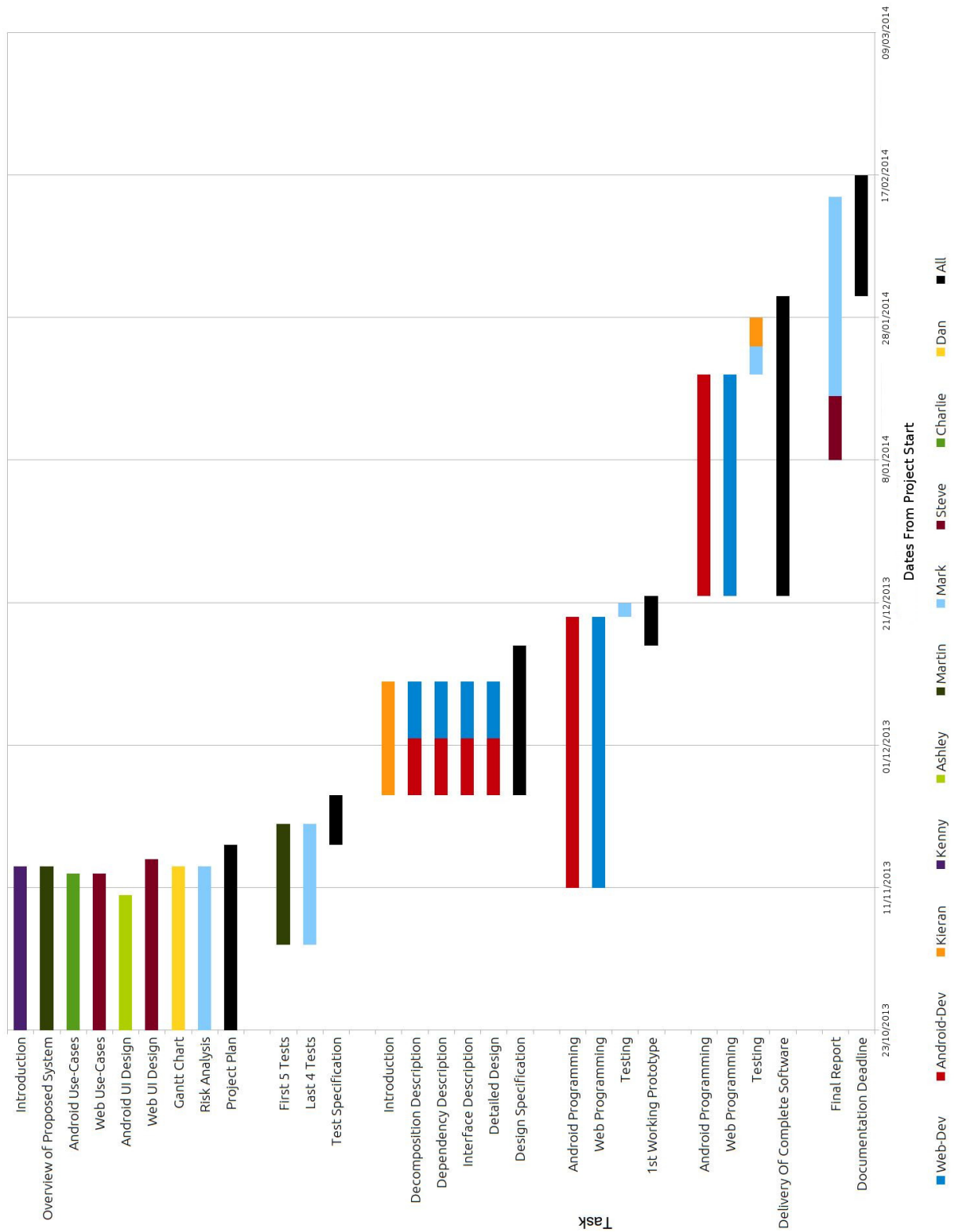
3.2.4 APIs

A large part of the web interface's codebase will revolve around generating a map with the Google Maps API. The API will allow us to dynamically generate a map containing the waypoints for each route.

3.2.5 Android Version

The Android SDK version we are using is 8. This will allow our application to run on any android devices running version 2.2 or higher.

4 GANTT CHART



5 RISK ANALYSIS

5.1 Ongoing Tasks

Risk Event	L	M	Risk	Mitigation
Team member absence	0.6	0.3	0.18	All team members to regularly check emails and the agreed online resources for meeting times. Being unaware of meetings is not a valid excuse. If a team member is unable to attend a meeting, the project lead (Daniel) must be notified as soon as they know they can't attend.
Project Lead absence	0.6	0.5	0.30	Meeting to go ahead as planned with Charlie (Deputy Project Lead) taking the meeting.
QA Manager absence	0.6	0.3	0.18	Meeting to go ahead as planned with Steve (Deputy QA Manager).
Unable to contact team member	0.3	0.8	0.24	Ensure that all team members regularly check emails and other agreed online resources, as well as checking meeting minutes so they are aware of any outstanding tasks/actions. Persistently being unreliable with result in a warning, and further action if necessary; e.g. carding or role reallocation.
Git failure	0.3	1.0	0.3	All work to be backed up regularly in several places in case of human error or Git failure.
Major illness or unexpected circumstances	0.5	0.9	0.45	Team members to be notified as soon as possible, in case any urgent tasks need to be re-assigned or completed by another team member.
Git conflict	0.3	0.9	0.27	All team members are to have read the information on the project wiki on Git conflicts. If a conflict is encountered, then it must be resolved immediately. If a conflict cannot be resolved easily, then an appropriate team member (Git expert/project lead) must be notified and the conflict must be resolved. Try to ensure an even task allocation, to avoid multiple team members working on the same code simultaneously.

5.2 Documentation

Risk Event	L	M	Risk	Mitigation
Late submission	0.4	0.8	0.32	Deadlines for documentation to be brought forward to ensure that any future problems encountered will come to light within a reasonable time frame. If team members run into any problems, they are to alert the group so that a solution can be issued.
Inadequate quality submission	0.4	1.0	0.4	Documents to be checked by either of the QA managers or project leaders before submitting. If team members need help then they should ask the rest of the team for help.
Human error	0.5	0.2	0.1	All documents to be checked for spelling, grammar, logic, and clerical errors by the creator of each document and at least one other team member.
Loss of documentation	0.4	1.0	0.4	Documentation to be stored and versioned on Git and backed up individually by the document's creator. Each individual is responsible for their own documentation.

5.3 Software Development

Risk Event	L	M	Risk	Mitigation
Project behind schedule	0.5	1.0	0.5	Development to begin as soon as possible. Charlie (lead developer) to delegate any appropriate outstanding tasks. Regular feedback to be given by Daniel (project lead) to ensure schedule is adhered to.
Parts of the project missing/incomplete	0.4	1.0	0.4	Daniel (project lead) and Charlie (lead developer) to delegate programming tasks appropriately. Constant testing by QA managers and project leaders to check if code is of a sufficient quality.

5.4 Risk Grade and Recommended Action Key

Risk Grade	Less than negligible	Negligible	Acute	Severe	Critical	Catastrophic
Risk score	< 0.2	0.2 - 0.39	0.4 - 0.59	0.6 - 0.79	0.8 - 0.99	1.0
Action	Tolerate	Tolerate	Tolerate or treat	Treat	Transfer	Terminate

6 USE CASES AND SCENARIOS

6.1 Overview of Use Cases

6.1.1 Core Functionality

The "core functionality" sections represent the highest priority features for the project, and will be the bare minimum functionality that the project will contain.

6.1.2 Extended Functionality

The "extended functionality" sections represent the lower priority features, that will be implemented within the project if time permits.

6.2 Android - Core Functionality

6.2.1 Create Route

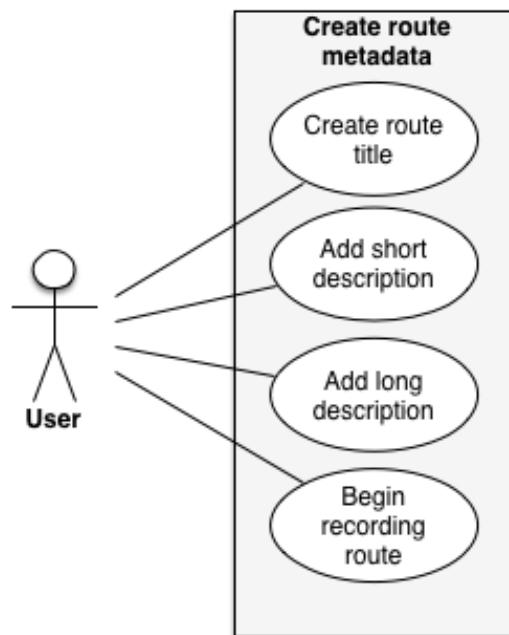


Figure 2: A user wants to create a new walk. The user will progress from the main screen to the "create route" screen, and fill in the route title, a short description (a tagline), and a long description. From there, the user can progress to recording the route.

6.2.2 Record Route

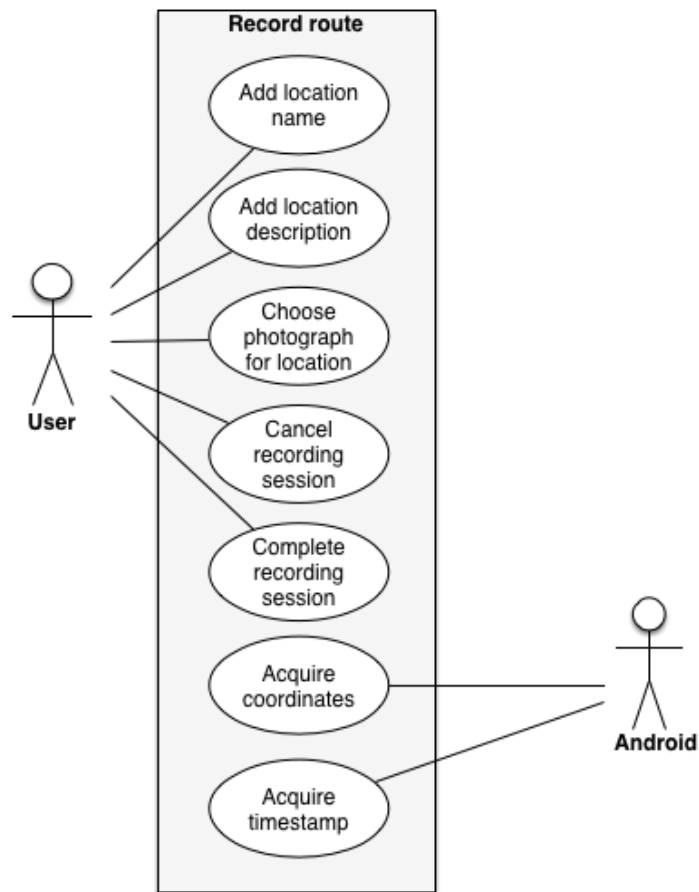


Figure 3: At each location (or waypoint) that the user chooses, they will retrieve their phone and enter the name and a description for the location. A photo can also be chosen from the gallery or taken using the camera - and then added to the waypoint. From this screen, the user can choose to complete the recording session, or cancel it entirely. The Android device will automate the task of fetching GPS coordinates and attaching a timestamp to each entry.

6.2.3 Complete Route

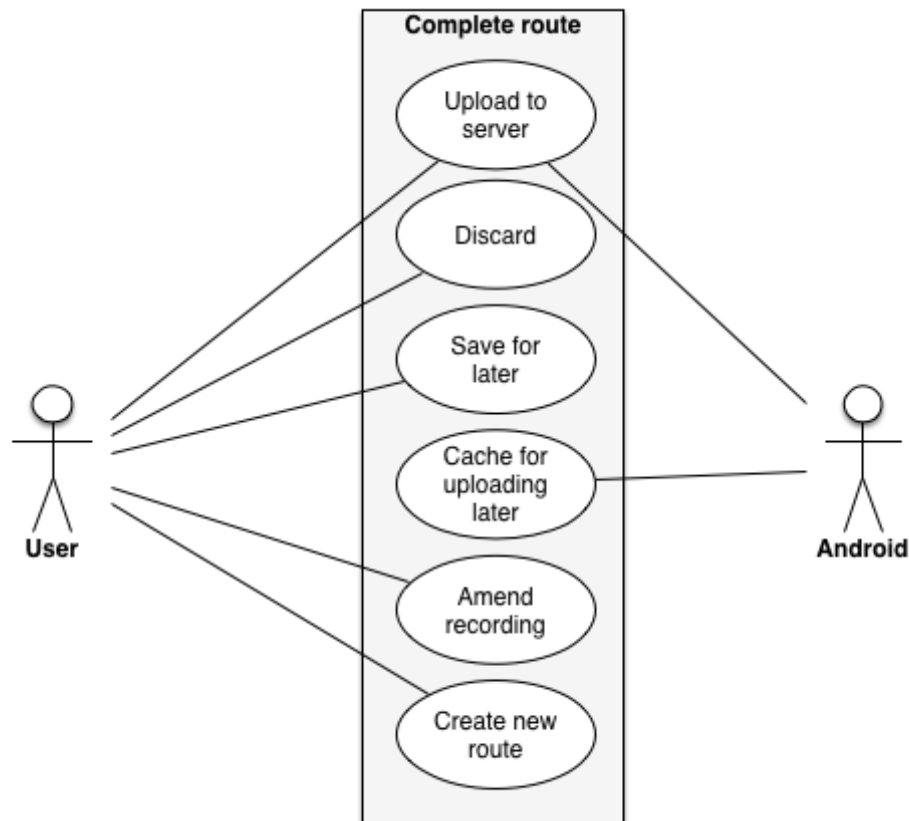


Figure 4: After the user has entered all of the locations that they wish to add and elected to complete the recording, they will be presented with several choices. They can upload the changes to the server (this will be handled by the Android system), saving the walk for later, discarding the walk entirely, or amending the recording - in case a mistake was made. The Android system will also be able to cache the recording for later upload (in the case of no signal, or other upload errors). The user will also be able to create a new route from this screen.

6.2.4 System

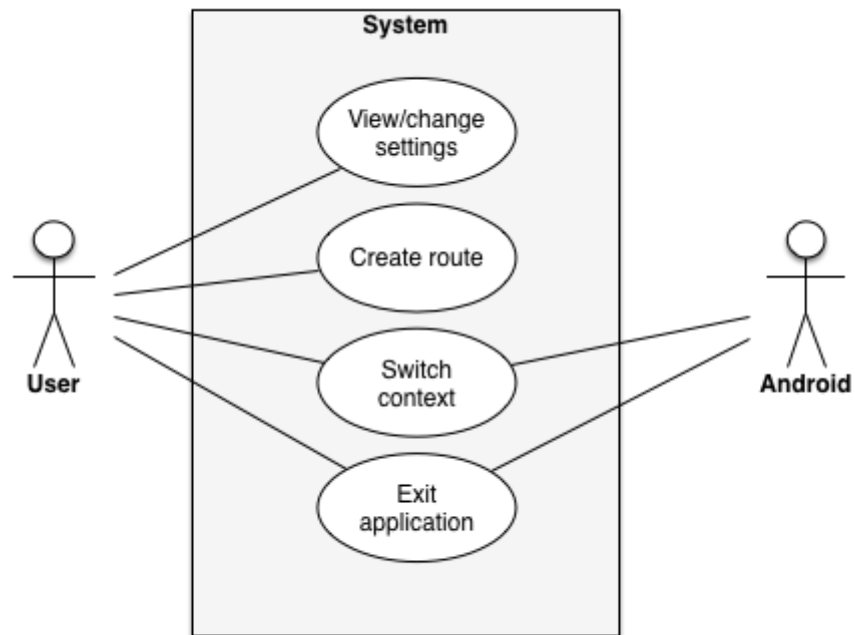


Figure 5: The user will be able to view or change their default settings and create a route from this screen. The application will also be able to handle context switching (i.e. switching to another app) and the application will be able to handle exits safely.

6.3 Android - Extended Functionality

6.3.1 Create Route

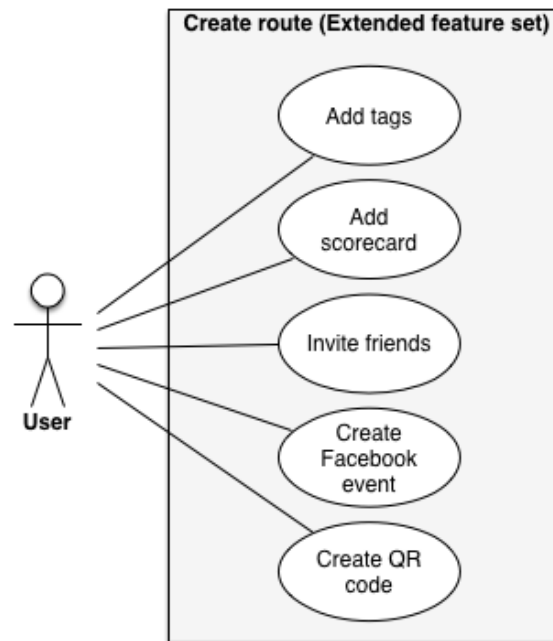


Figure 6: The user will be able to 'tag' their walk(s) with keywords to aid searching on the website. The user will be able to create a Facebook event (through a Facebook app), and invite friends to such an event. Another function available would be to generate a QR code to the Facebook event URL, to help the user share the event effectively.

6.3.2 Record Route

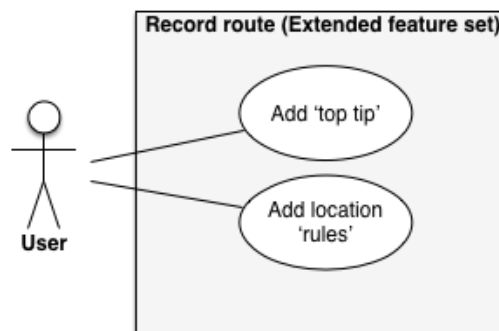


Figure 7: The user will be able to add a 'top tip' for every location; for example, a favourite drink or best place to sit. The user will also be able to choose 'rules' for each location for the followers of the walk - so as to allow the participants of the walk to play a game, or complete a challenge.

6.3.3 Complete Route

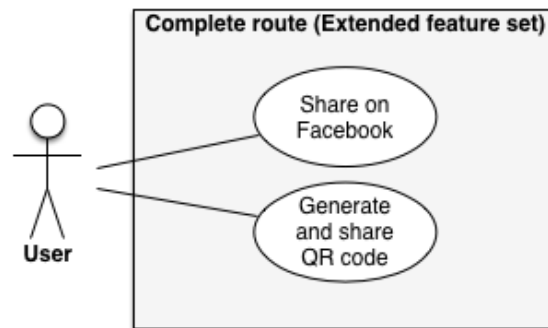


Figure 8: The user will be able to choose to share the create event on Facebook using a Facebook app, as well as sharing the QR code generated earlier.

6.3.4 System

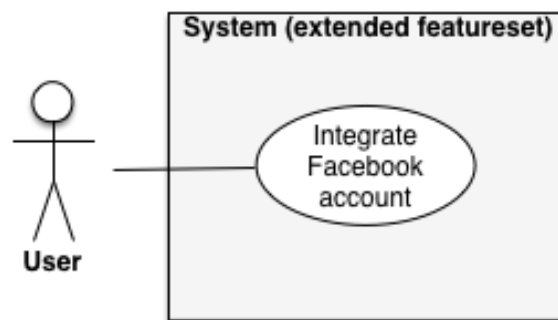


Figure 9: The user will be able to integrate the application with their Facebook account (using a Facebook app and OAuth authentication).

6.4 Website - Core Functionality

6.4.1 System

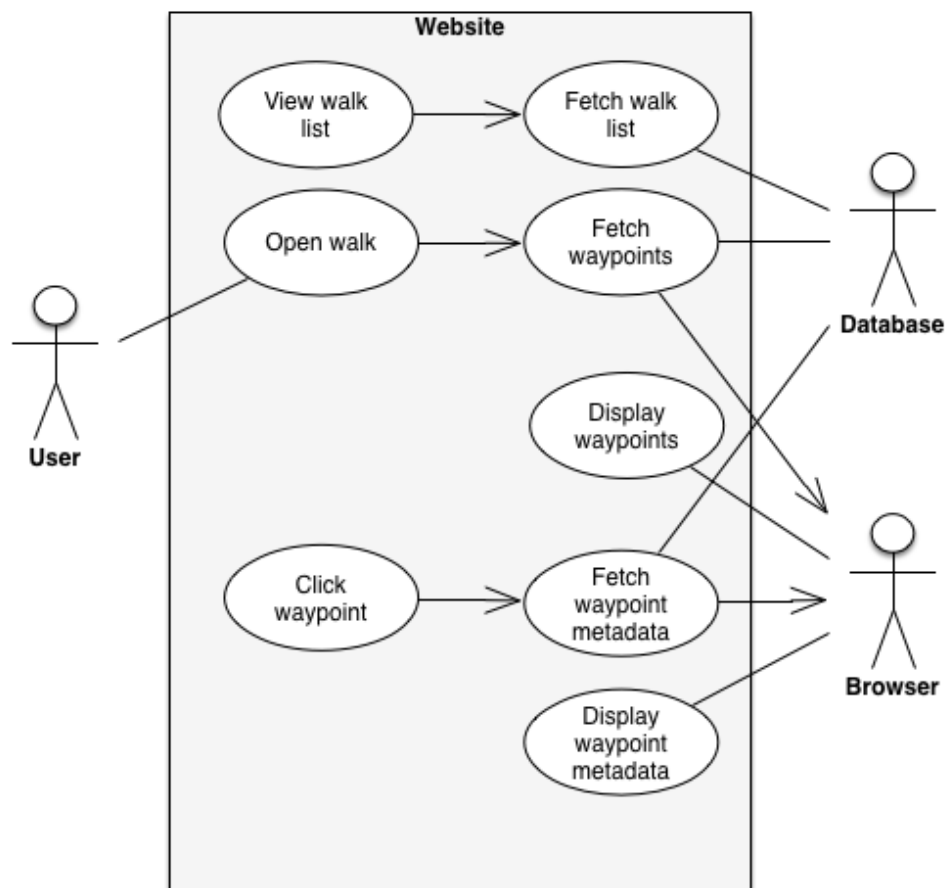


Figure 10: When the page loads, the user will be prompted with a list of walks to load. This will be achieved by the webpage querying the database - the database will provide the information and display it to the user in the browser. The user will be able to open a walk using the web interface, and this will request the walk's information from the database and cause the browser to display the information for each walk on a map. Upon the user clicking a waypoint on the map, the information for each waypoint will be fetched from the database and displayed on-screen.

6.5 Website - Extended Functionality

6.5.1 System

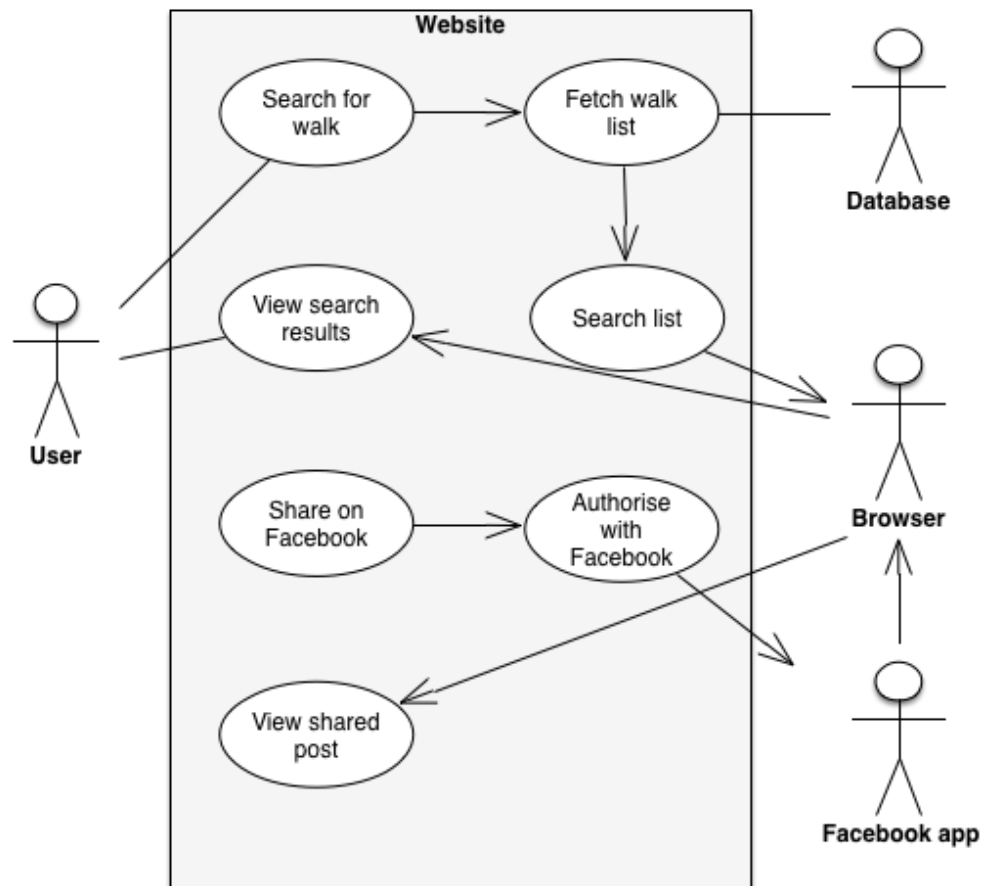


Figure 11: The user will be able to search for walks within the browser, which will search through the database and display the matching walks. The user will also be able to share walks on Facebook using a Facebook app.

7 USER INTERFACE DESIGNS

7.1 Android Interface Designs

7.1.1 Home

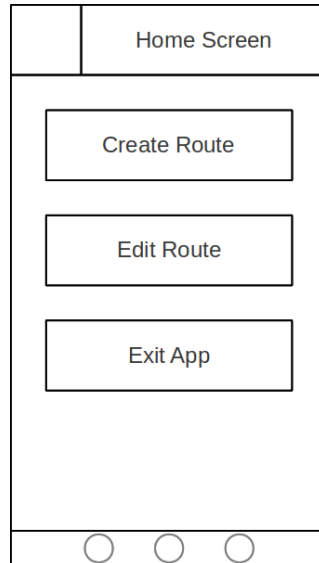


Figure 12: The home screen is the first screen that the user will be able to see when the app is loaded. There are three buttons on this page the first two will, when clicked, take the user to the Route Info screen. The last button on this screen will close the application.

7.1.2 Create Route

Route Info	
Title : _____	
<div>Short description</div>	
<div>Long description</div>	
<div>Next</div>	

Figure 13: When the user wants to create a new route this is the next screen that they will encounter. The three text boxes will allow the user to enter the route's identifier (title) as well as descriptive information about the route. At the bottom of the screen is the next button that will allow the user to progress the the Route Content screen.

7.1.3 Add Waypoint

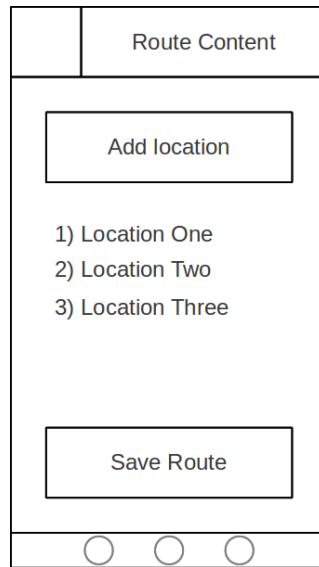


Figure 14: On this screen the user can add locations of interest to the route by clicking on the Add Location button, this action will also take the user to the Location Info screen. After a location has been added it will appear in the list of locations underneath the Add Location button. When the user is finished adding locations to the route they can click on the Save Route button, at this point they will be prompted as to whether they wish to upload to the server or save the route locally for editing later.

7.1.4 Location Information

The image shows a mobile application interface for 'Location Info'. It features a title bar at the top. Below the title bar, there are two input fields: 'Name : _____' and 'Description' followed by a text area. Below the text area is a button labeled 'Add image'. Underneath the button is a large rectangular placeholder for an image, with a small square button containing an 'X' in the top right corner. At the bottom of the screen is a navigation bar with three circular icons.

Figure 15: The name and description can be entered into text boxes on this screen. Also images of the location can be added using the Add Image button. Images that have been added will be displayed below the Add Image button, and below the image place holder is the Save button that will return the user to the Route Content screen.

7.2 Web Interface Designs

7.2.1 Main Web Interface

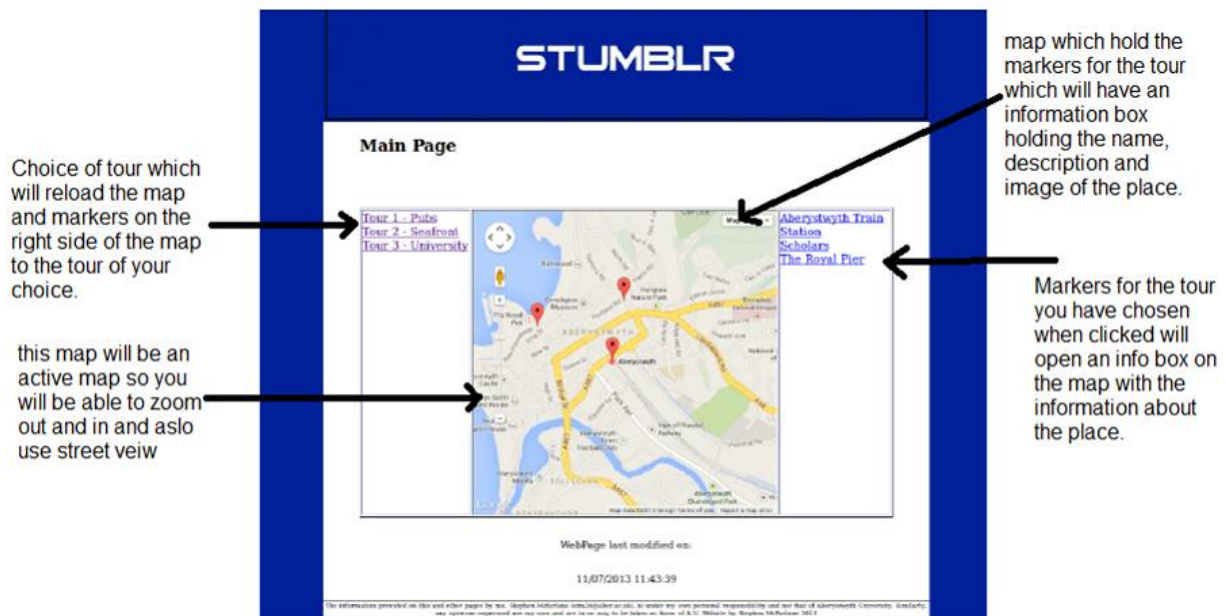


Figure 16: This is a sample UI for the tour viewer; The design of the final page might be different.

8 REFERENCES

- [1] PAVLIC, T. Programming Assignment template for L^AT_EX.
<http://www.latextemplates.com/template/programming-coding-assignment>.
Accessed: Oct 23, 2013.
- [2] TIDDEMAN, B. P. Software Engineering Group Projects - Project Plan Specification Standards.

9 VERSION HISTORY

Author	Date	Version	Change made	CCF
CCN	07/11/2013	1.1	Updated order of authors on cover	n/a
DEC	07/11/2013	1.2	Added High Level Architecture diagram	#37

Group 10 Design Specification

February 15, 2014

SE.10.D3

Version: 1.6

Status: Release

Contributor Name	Role
Daniel Clark	Project Lead
Mark Lewis	QA Manager
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

©Copyright Group 10, 2013

Contents

1 INTRODUCTION	4
1.1 Purpose of This Document	4
1.2 Scope	4
1.3 Objectives	4
2 DECOMPOSITION DESCRIPTION	4
2.1 Programs in System	4
2.2 Significant Classes - Android	5
2.2.1 StumblrData	5
2.2.2 Waypoint	5
2.2.3 Route	5
2.3 Significant Activities - Android	5
2.3.1 AbstractActivity	5
2.3.2 CreateWaypoint	5
2.3.3 CreateRoute	5
2.3.4 GPSService	5
2.3.5 WaypointList	5
2.3.6 FinishRoute	5
2.3.7 Home	6
2.4 Significant Classes - Web	6
2.4.1 Page	6
2.4.2 Database Interaction	6
2.5 Shared Modules Between Programs	6
2.6 Mapping Requirements to Classes	7
3 DEPENDENCY DESCRIPTION	8
3.1 Class Diagrams - Inheritance Relationships	8
3.1.1 Android Activity Classes	8
3.1.2 Android Data Classes	9
4 INTERFACE DESCRIPTION	10
4.1 Android Activities	10
4.1.1 AbstractActivity {abstract}	10
4.1.2 Home	12
4.1.3 CreateRoute	13
4.1.4 CreateWaypoint	15
4.1.5 GPSService	18
4.1.6 WaypointList	21
4.1.7 FinishRoute	25
4.2 Android Data Classes	27
4.2.1 StumblrData {abstract}	27
4.2.2 Route	29
4.2.3 Waypoint	33
4.3 Web Interface Descriptions	36
4.3.1 Index.php	36
5 DETAILED DESIGN	37
5.1 Sequence Diagram	37
5.2 Significant Algorithms	38
5.2.1 Upload Data	38
5.2.2 Bundle Data	38
5.2.3 Validate Inputs	38
5.2.4 Change Screen	38

5.3	Significant Data Structures - Android	38
5.3.1	Route	38
5.3.2	Waypoint	38
5.3.3	Co-ordinates	38
5.4	Significant Data Structures - MySQL Database	39
5.4.1	List of Walks	39
5.4.2	Location	39
5.4.3	Place Description	39
5.4.4	Photo Usage	39
5.5	Website Interface Specification	39
5.5.1	Page	39
5.5.2	Map	40
5.5.3	Database Interaction	40
5.6	JSON Example	41
5.7	HTTP Transaction and Expected Values	41
6	Android Design Justifications	42
7	Web Design Justifications	42
8	REFERENCES	43
9	VERSION HISTORY	43

1 INTRODUCTION

1.1 Purpose of This Document

The purpose of this document is to ensure that the developers have a complete and holistic understanding of how the project is going to be structured and evolve into a fully functional system. This will be done by establishing the key aspects and identifying each unique requirement stated initially by the client. Following are the key aspects such as descriptions of classes and how the interface is going to be presented to the user and to specify the procedures taken to ensure their completion.

1.2 Scope

This design specification will separate the project into components which are implemented separately from each other - describing the high-level interactions between components.

1.3 Objectives

- To provide a description of the main components of Stumblr.
- To provide an explanation of the dependencies between certain components, describing for both compilation and execution.
- To provide details about the interface for all main classes of Stumblr.

2 DECOMPOSITION DESCRIPTION

2.1 Programs in System

The System that will be created will contain three systems; the Android app, a server and a web page.

The Android side of the system will be used as the platform for the walking tour creator app. Within this app users can create a new tour, they will then be able to add locations to the tour. These locations will contain information such as the tours title, description, the way points of the location and possibly an image. Once the tour is saved and finished it can then be sent via Wi-Fi or internet connection on the device to the server. The data will be sent to the server as a Multimedia Internet Message Extension (MIME) format in an HTTP POST request.

The server will be used to store all data about tours created on the Android app. On the server a database will be created and within the database a table for storing the tour's details. Once data is received a new record will be created and the data stored for easy re-use on the web page. To access this data the web page will use a PHP script with MySQL commands.

The final program in the system is the web page, which will be used for displaying tours. On the website there will be a list of saved tours from which the user can click to view them. The tour will then be displayed on a map which will incorporate the use of the Google Maps API. By clicking on a Waypoint the user will be able to view the information added by the creator.

2.2 Significant Classes - Android

2.2.1 StumblrData

This will be an abstract class which will contain variables and validation methods that other classes will use, such as "short description" and "title".

2.2.2 Waypoint

This class will handle all information and methods regarding each individual Waypoint on the route, including acquisition and processing of image and GPS data from the Android system. A constructor method will be used to create a Waypoint containing "title", "short description" and an image. Coordinates will be stored in a LinkedList of (android.location.Location) objects. The LinkedList will contain the list of coordinates between the current and previous Waypoints.

2.2.3 Route

This class will contain all the methods to handle all parts of creating a route, such as adding Waypoints to a linked list of Waypoints.

2.3 Significant Activities - Android

2.3.1 AbstractActivity

AbstractActivity will hold all of the common methods (common bundle loading/saving techniques). Each other activity will inherit from it.

2.3.2 CreateWaypoint

Inherits from DataEntryActivity. Contains methods to set images, timestamps, GPS coordinates, and other information in the constructor.

2.3.3 CreateRoute

Inherits from DataEntryActivity. Contains methods to create a new Route LinkedList and access items within the LinkedList. Also contains methods to access data held within text fields - from the corresponding data entry field.

2.3.4 GPSService

A simple foreground service that serves as a way of obtaining GPS coordinates without being killed by the Android system.

2.3.5 WaypointList

Inherits from DataEntryActivity. This activity contains methods and UI elements to show the Waypoint objects in a list-type structure within the app.

2.3.6 FinishRoute

Inherits from DataEntryActivity. This contains methods to package the Route object into something that can be uploaded (via JSON) to the server application, as well as giving the user a persistent notification icon showing the status of their upload.

2.3.7 Home

Inherits from DataEntryActivity. This is the main screen of the application, that shows all of the menu options necessary (and also the Settings menu). This will contain methods to progress to the CreateRoute activity.

2.4 Significant Classes - Web

2.4.1 Page

The main web interface will make use of the Map and the Database Interaction files. The main interface will employ JavaScript, HTML, and CSS. This page will show the map, using the database interaction file to pull the relevant information from the MySQL database, and render it on-screen using the Google Maps API and JavaScript.

2.4.2 Database Interaction

The database interaction file will abstract the database interaction away from the main web interface page; it will be used to add and request data from the MySQL database. Included in the database will be the locations for the Maps API, as well as showing different tours in a table next to it markers for each route's Waypoints in another table.

2.5 Shared Modules Between Programs

As the two programs differ enormously in terms of the technology used, there are no common modules, per se. However, one could consider the fact that both programs are expected to communicate using the HTTP protocol.

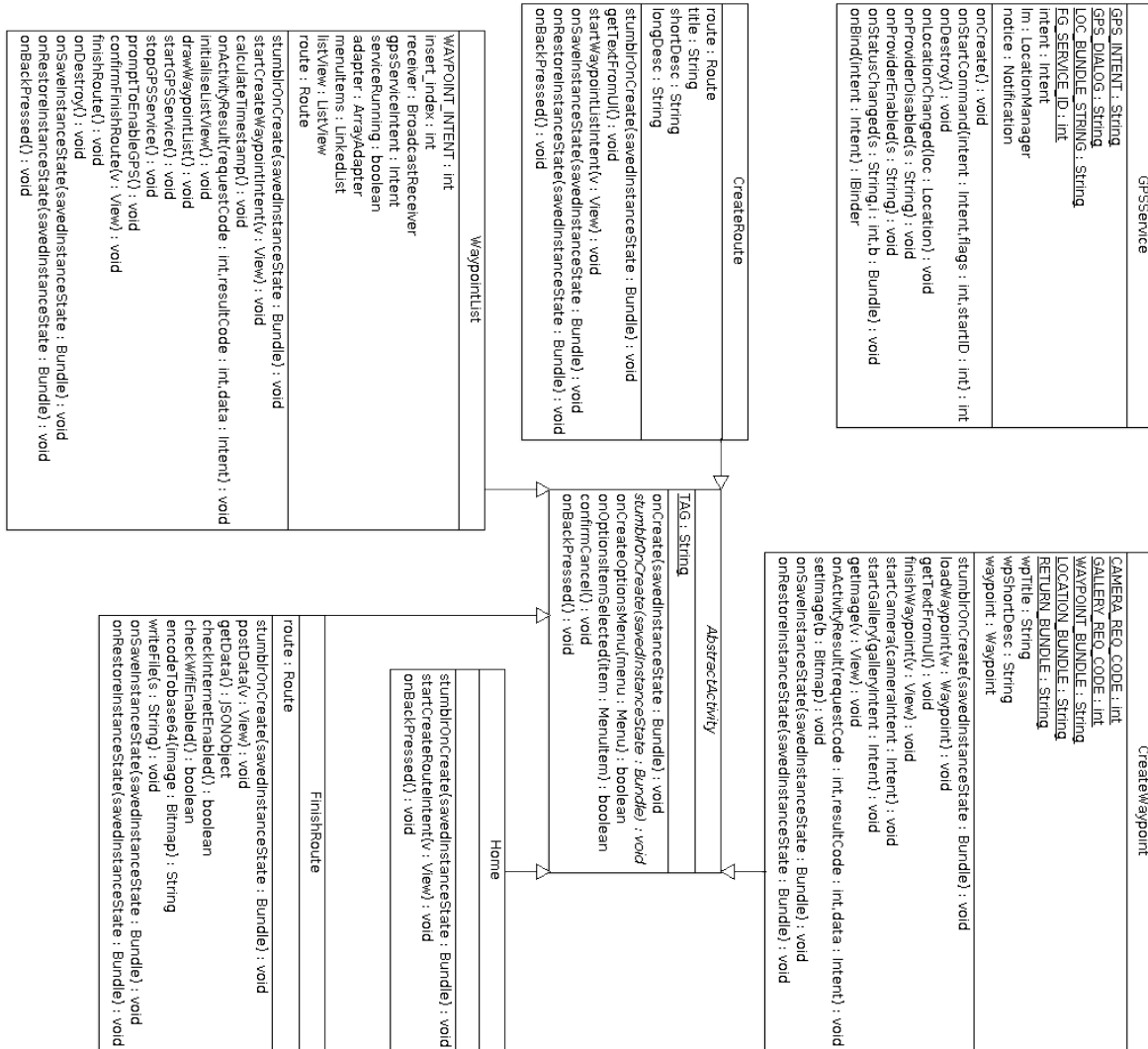
2.6 Mapping Requirements to Classes

1. **FR1; Startup of software on Android device**
Android class Home and Android class AbstractActivity.
2. **FR2; Providing info about the whole walking tour**
Android class WaypointList and AbstractActivity.
3. **FR3; Adding locations to the walking tour**
Android class CreateWaypoint, and Android class AbstractActivity.
4. **FR4; Adding photos to the walks**
Android class CreateWaypoint and Android class AbstractActivity.
5. **FR5; Cancelling walks**
Android class CreateWaypoint and Android class AbstractActivity.
6. **FR6; Sending the walk to the server**
Android class FinishRoute and Android class AbstractActivity.
7. **FR7; Switching from the WTC**
Android class AbstractActivity holds the basic code for switching between apps. Each individual Activity class contains its own code in **onSavedInstanceState()** methods.
8. **FR8; Trying out a created walking tour**
Web Page, Map and Database Interaction.
9. **FR9; Saving data on server**
Android class FinishRoute, Android class AbstractActivity, Web Page, Map and Database Interaction.

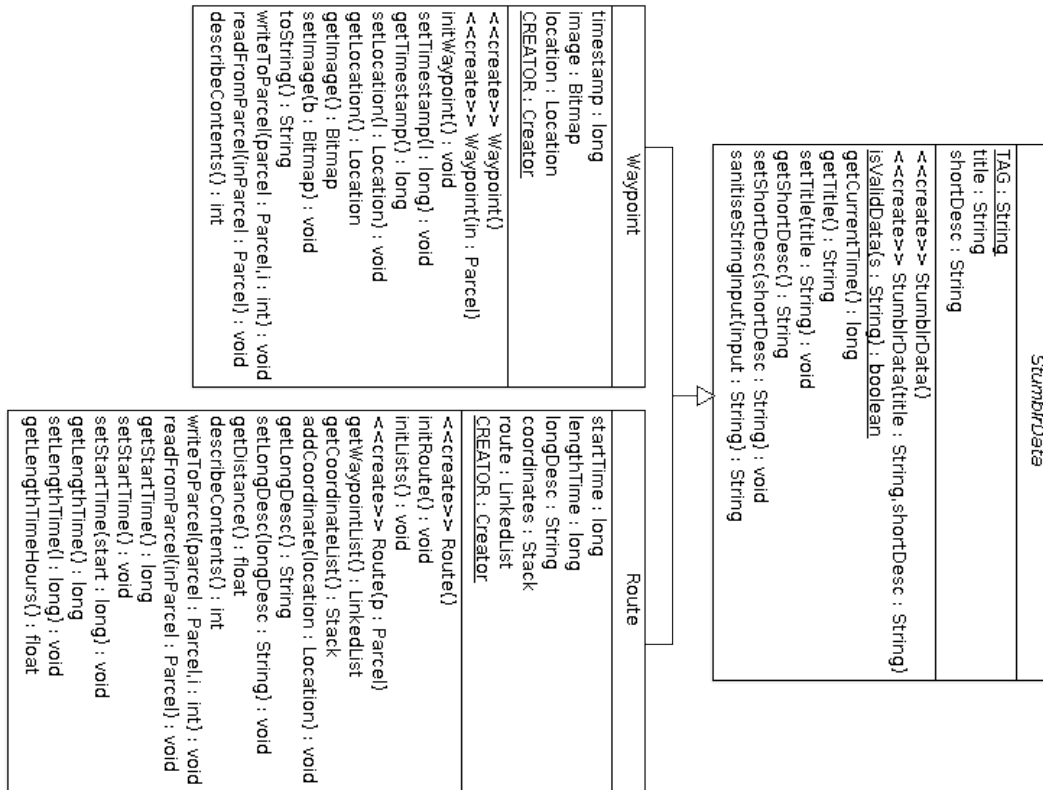
3 DEPENDENCY DESCRIPTION

3.1 Class Diagrams - Inheritance Relationships

3.1.1 Android Activity Classes



3.1.2 Android Data Classes



4 INTERFACE DESCRIPTION

4.1 Android Activities

4.1.1 AbstractActivity {abstract}

Listing 1: AbstractActivity {Abstract}

```
1 package uk.ac.aber.cs.groupten.stumblr;

3 import android.app.AlertDialog;
import android.content.DialogInterface;
5 import android.content.Intent;
import android.os.Bundle;
7 import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
9 import android.view.MenuItem;

11 public abstract class AbstractActivity extends ActionBarActivity {
    /**
13     * Logging tag.
    */
15     public static final String TAG = "STUMBLR";

17     /**
    * @param savedInstanceState saved state of the application.
19     * onCreate set state to saved previously frozen state.
    */
21     @Override
    protected void onCreate(Bundle savedInstanceState);

23
25     /**
    * @param savedInstanceState
    * Called by super class
27     */
    public abstract void stumblrOnCreate(Bundle savedInstanceState);

29
31     /**
    * @param menu inflate the menu; this adds items to the action bar if
    it is present.
    * @return true once menu has inflated.
33     */
    @Override
35     public boolean onCreateOptionsMenu(Menu menu);

37
39     /**
    * Handle action bar item clicks here. The action bar will
    automatically handle
    * clicks on the Home/Up button, so long as you specify a parent
    activity in
    * AndroidManifest.xml
41     * @param item passed in MenuItem when it is selected
    * @return selected item
43     */
    @Override
```



```
45     public boolean onOptionsItemSelected(MenuItem item);
47     public void confirmCancel();
49     /**
    * When back button is pressed, call finish. Drop activity from memory
    *
    */
51     @Override
53     public void onBackPressed();
}
```

4.1.2 Home

Listing 2: Home

```
package uk.ac.aber.cs.groupten.stumblr;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Home extends AbstractActivity {
    /**
     * Loads the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
     * state.
     */
    public void stumblrOnCreate(Bundle savedInstanceState);

    /**
     * Begin the Route entry activity
     */
    public void startCreateRouteIntent(View v);

    /**
     * When back is pressed, finish activity.
     */
    @Override
    public void onBackPressed();
}
```

4.1.3 CreateRoute

Listing 3: CreateRoute

```
package uk.ac.aber.cs.groupten.stumblr;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import uk.ac.aber.cs.groupten.stumblr.data.Route;
import uk.ac.aber.cs.groupten.stumblr.data.StumblrData;

public class CreateRoute extends AbstractActivity {

    /**
     * Instance of Route.
     */
    private Route route;

    /**
     * Title of route.
     */
    private String title;

    /**
     * Short description of route.
     */
    private String shortDesc;

    /**
     * Long description of route.
     */
    private String longDesc;

    /**
     * Loads the activity on creation (using a bundle if one is present)
     *
     * @param savedInstanceState The bundle containing the saved instance
     *        state.
     */
    @Override
    public void stumblrOnCreate(Bundle savedInstanceState);

    /**
     * Get text from fields in the user interface and sanitise inputs.
     */
    public void getTextFromUI();

    /**
     * Start the WaypointList activity (list the current Route).
     * Called when the "next" button is clicked in the UI.
     */
}
```

```
52     */
53     public void startWaypointListIntent(View v);
54
55     /**
56      * On instance saved, call getTextFromUI() and set the strings to
57      * appropriate variables.
58      * @param savedInstanceState
59      */
60     @Override
61     public void onSaveInstanceState(Bundle savedInstanceState);
62
63     /**
64      * Restore instance state to savedInstanceState. Set title, shortDesc
65      * and longDesc by getting
66      * them from the savedInstanceState. Then set textview fields to their
67      * appropriate variables.
68      * @param savedInstanceState
69      */
70     @Override
71     public void onRestoreInstanceState(Bundle savedInstanceState);
72
73     /**
74      * Logs when back is pressed in CreateRoute.
75      */
76     @Override
77     public void onBackPressed()
78 }
79 }
```

4.1.4 CreateWaypoint

Listing 4: CreateWaypoint

```
package uk.ac.aber.cs.groupten.stumblr;

2
import android.app.AlertDialog;
4 import android.content.DialogInterface;
import android.content.Intent;
6 import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
8 import android.location.Location;
import android.net.Uri;
10 import android.os.Bundle;
import android.provider.MediaStore;
12 import android.util.Log;
import android.view.View;
14 import android.view.WindowManager;
import android.widget.ImageView;
16 import android.widget.TextView;
import android.widget.Toast;

18
import java.io.FileNotFoundException;
20 import java.io.IOException;
import java.io.IOException;
22 import java.util.Calendar;

24 import uk.ac.aber.cs.groupten.stumblr.data.StumblrData;
import uk.ac.aber.cs.groupten.stumblr.data.Waypoint;
26
public class CreateWaypoint extends AbstractActivity {
28     /**
     * Request code for camera intent
30     */
    public static final int CAMERA_REQ_CODE = 1337;
32
    /**
34     * Request code for gallery intent
     */
36     public static final int GALLERY_REQ_CODE = 7007;

38     /**
     * Bundle string for identifying Waypoint data
40     */
    public static final String WAYPOINT_BUNDLE = "waypoint";
42
    /**
44     * Bundle string for identifying location data
     */
46     public static final String LOCATION_BUNDLE = "loc";

48     /**
     * Bundle string for identifying return data
50     */
    public static final String RETURN_BUNDLE = "return_data";
```

```
52
    /**
54     * Waypoint title.
    */
56     private String wpTitle;

58     /**
    * Short description.
60     */
    private String wpShortDesc;

62
    /**
64     * Instance of waypoint.
    */
66     private Waypoint waypoint;

68     /**
    * Loads the activity on creation (using a bundle if one is present)
70     *
    * @param savedInstanceState The bundle containing the saved instance
    *       state.
72     */
    public void stumblrOnCreate(Bundle savedInstanceState);

74
    /**
76     * Pass in waypoint. Set TextView and ImageView fields to data
    *   returned from getting
    * the waypoints title, short description and image
78     * @param w passed in waypoint to be loaded.
    */
80     public void loadWaypoint(Waypoint w);

82
    /**
    * Gets waypoint title and short description from the user interface.
84     */
    public void getTextFromUI();

86
    /**
88     * Called when "Create" button in the UI is clicked.
    * Adds data to the current Waypoint object with text specified in UI.
90     */
    public void finishWaypoint(View v);

92
    /**
94     * *****
    *                               Camera interaction                               *
96     * *****
    */
98     /**
    * Obtain a photo from user and add it to current Waypoint.
100     */
    public void startCamera(Intent cameraIntent);

102
    public void startGallery(Intent galleryIntent);
```

```
104     public void getImage(View v);
106
107     /**
108      * on Creation of a waypoint result, if the result is OK then
109      *   getExtras from intent data.
110      * Then get bitmap image for waypoint image and check for null prior
111      * to setting the image to the
112      * waypoint. Update UI imageView, and log dimensions of image.
113      *
114      * @param requestCode
115      * @param resultCode
116      * @param data
117      */
118     @Override
119     protected void onActivityResult(int requestCode, int resultCode,
120         Intent data);
121
122     public void setImage(Bitmap b);
123
124     @Override
125     public void onSaveInstanceState(Bundle savedInstanceState);
126
127     /**
128      * Restore state to savedInstanceState. Set waypoint title, short
129      * description and waypoint
130      * to appropriate data retrieved from the savedInstanceState. Then
131      * update TextViews with
132      * newly set title and short description.
133      * @param savedInstanceState
134      */
135     @Override
136     public void onRestoreInstanceState(Bundle savedInstanceState);
137 }
```

4.1.5 GPSService

Listing 5: DataEntryActivity {Abstract}

```
package uk.ac.aber.cs.groupten.stumblr;

2
import android.app.Notification;
4 import android.app.PendingIntent;
import android.app.Service;
6 import android.content.Context;
import android.content.Intent;
8 import android.location.Location;
import android.location.LocationListener;
10 import android.location.LocationManager;
import android.os.Build;
12 import android.os.Bundle;
import android.os.IBinder;
14 import android.support.v4.app.NotificationCompat;

16 public class GPSService extends Service implements LocationListener {

18     /**
19      * String name for GPS_INTENT.
20     */
21     public final static String GPS_INTENT = "STUMBLR_GPS";
22
23     /**
24      * String name for GPS_DIALOG.
25     */
26     public final static String GPS_DIALOG = "STUMBLR_GPS_DIALOG";
27
28     /**
29      * String name for bundle of location strings.
30     */
31     public final static String LOC_BUNDLE_STRING = "loc";
32
33     /**
34      * Service ID.
35     */
36     private static final int FG_SERVICE_ID = 101;
37
38     /**
39      * Instance of Intent.
40     */
41     private Intent intent;
42
43     /**
44      * Instance of LocationManager.
45     */
46     private LocationManager lm;
47
48     /**
49      * Instance of Notification.
50     */
51     private Notification notice;
```



```
52
    /**
54     * onCreate start new intent for GPS_INTENT
    */
56     @Override
    public void onCreate();
58
    /**
60     * @param intent The intent that the Service was started from
    * @param flags Startup flags
62     * @param startID Service ID
    * @return The service status (Sticky, non-sticky, etc)
64     */
    @Override
66     public int onStartCommand(Intent intent, int flags, int startID);
68
    /**
    * onDestroy remove LocationManager updates and stop foreground task.
70     */
    @Override
72     public void onDestroy();
74
    /**
    * Obtain coordinates from Android system and add to current Waypoint.
76     */
    @Override
78     public void onLocationChanged(Location loc);
80
    /**
    * Prompt for GPS, and error handling
82     */
    @Override
84     public void onProviderDisabled(String s);
86
    /**
    * Empty method to abide by implementation requirements.
88     * @param s
    */
    @Override
90     public void onProviderEnabled(String s);
92
    /**
94     * Empty method to abide by implementation requirements.
    * @param s
96     * @param i
    * @param b
98     */
    @Override
100     public void onStatusChanged(String s, int i, Bundle b);
102
    /**
    * Empty method to abide by implementation requirements.
104     * @param intent
    * @return
```

```
106      */  
      public IBinder onBind(Intent intent); // Unused  
108  }
```

4.1.6 WaypointList

Listing 6: WaypointList

```
package uk.ac.aber.cs.groupten.stumblr;

2
import android.app.AlertDialog;
4 import android.app.Dialog;
import android.content.BroadcastReceiver;
6 import android.content.Context;
import android.content.DialogInterface;
8 import android.content.Intent;
import android.content.IntentFilter;
10 import android.location.Location;
import android.os.Bundle;
12 import android.provider.Settings;
import android.util.Log;
14 import android.view.View;
import android.widget.AdapterView;
16 import android.widget.ArrayAdapter;
import android.widget.ListView;
18
import java.util.EmptyStackException;
20 import java.util.LinkedList;

22 import uk.ac.aber.cs.groupten.stumblr.data.Route;
import uk.ac.aber.cs.groupten.stumblr.data.Waypoint;
24
public class WaypointList extends AbstractActivity {
26
    /**
28     * Waypoint intent constant.
    */
30     private final int WAYPOINT_INTENT = 3141;

32     /**
    * Index to insert new waypoint.
    */
34     private int insert_index;

36     /**
38     * GPS Broadcast reciever.
    */
40     private BroadcastReceiver receiver;

42     /**
    * GPS Service intent.
    */
44     private Intent gpsServiceIntent;

46     /**
48     * Boolean to show if GPS service is running.
    */
50     private boolean serviceRunning = false;
```

```
52     // ListView objects
53     /**
54      * Adapter used to put menu items into linked list
55      */
56     private ArrayAdapter<Waypoint> adapter;
57
58     /**
59      * List of menu items.
60      */
61     private LinkedList<Waypoint> menuItems;
62
63     /**
64      * View of the list.
65      */
66     private ListView listView;
67
68     /**
69      * Instance of route.
70      */
71     private Route route;
72
73     /**
74      * Loads the activity on creation (using a bundle if one is present)
75      *
76      * @param savedInstanceState The bundle containing the saved instance
77      *        state.
78      */
79     public void stumblrOnCreate(Bundle savedInstanceState);
80
81     /**
82      * Starts the CreateWaypoint Intent, so that it returns a result.
83      *
84      * @param v The View object.
85      */
86     public void startCreateWaypointIntent(View v);
87
88     /**
89      * Calculate time by subtracting the endTime by the startTime.
90      * Set the length route was recorded for to the route.
91      */
92     private void calculateTimestamp();
93
94     /**
95      * Called when an Activity is dispatched for a result
96      *
97      * @param requestCode Integer relating the intent to a request
98      * @param resultCode Used to check if a request was successful
99      * @param data The Intent used
100      */
101     @Override
102     protected void onActivityResult(int requestCode, int resultCode,
103                                     Intent data);
104
105     /**
```

```
104     * Initialises the ListView and Adapter objects.
105     */
106     private void initialiseListView();
107
108     /**
109     * Renders Waypoint list on screen.
110     */
111     public void drawWaypointList();
112
113     /**
114     * Starts the GPS service.
115     */
116     private void startGPSService();
117
118     /**
119     * Halts the GPS service.
120     */
121     private void stopGPSService();
122
123     /**
124     * Display dialog to prompt the user to enable GPS. Then take them to
125     * the settings page.
126     */
127     public void promptToEnableGPS();
128
129     /**
130     * Finish route method. Display message to check if the user is
131     * finished. If button is pressed
132     * call finishRoute and then finish the activity.
133     */
134     public void confirmFinishRoute(View v);
135
136     /**
137     * Passes the current Route object to FinishRoute and starts the
138     * activity.
139     * @param v The View object passed in by the Android OS.
140     */
141     public void finishRoute() {
142
143     /**
144     * onDestroy, stop the GPS service.
145     */
146     @Override
147     public void onDestroy();
148
149     /**
150     * When instance state is saved, put instance variables into it as
151     * well as route data.
152     * @param savedInstanceState
153     */
154     @Override
155     public void onSaveInstanceState(Bundle savedInstanceState);
156
157     /**
158     * When instance state is restored, retrieve instance variables and
```

```
154     * data from saved instance state.
155     * @param savedInstanceState
156     */
157     @Override
158     public void onRestoreInstanceState(Bundle savedInstanceState);

160     /**
161     * Log when back is pressed.
162     */
163     @Override
164     public void onBackPressed();
}
```

4.1.7 FinishRoute

Listing 7: FinishRoute

```
1 package uk.ac.aber.cs.groupten.stumblr;

3 import android.app.AlertDialog;
import android.app.Dialog;
5 import android.content.Context;
import android.content.DialogInterface;
7 import android.content.Intent;
import android.graphics.Bitmap;
9 import android.location.Location;
import android.net.ConnectivityManager;
11 import android.net.NetworkInfo;
import android.os.AsyncTask;
13 import android.os.Bundle;
import android.provider.Settings;
15 import android.util.Base64;
import android.util.Log;
17 import android.view.View;
import android.widget.TextView;
19 import android.widget.Toast;

21 import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
23 import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
25 import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
27 import org.json.JSONException;
import org.json.JSONObject;

29
import java.io.ByteArrayOutputStream;
31 import java.io.FileNotFoundException;
import java.io.FileOutputStream;
33 import java.io.IOException;
import java.io.OutputStreamWriter;
35 import java.lang.reflect.Method;
import java.util.LinkedList;
37 import java.util.Stack;

39 import uk.ac.aber.cs.groupten.stumblr.data.Route;
import uk.ac.aber.cs.groupten.stumblr.data.Waypoint;
41
43 public class FinishRoute extends AbstractActivity {
    private Route route;

45     /**
46      * Loads the activity on creation (using a bundle if one is present)
47      *
48      * @param savedInstanceState The bundle containing the saved instance
49      * state.
50      */
    public void stumblrOnCreate(Bundle savedInstanceState);
```

```
51      /*
52      * *****
53      *           HTTP POST Interaction           *
54      * *****
55      */
56      private class NetworkTask extends AsyncTask<String, Void, HttpResponse>
57      {
58          protected void onPostExecute(HttpResponse result); // Private
59              method
60
61      /**
62       * Posts the data to the server. Check if internet is available first.
63       */
64      public void postData(View v);
65
66      private JSONObject getData();
67      // End HTTP POST
68
69      /**
70       * Ensuring Network Provider is Enabled before submitting route
71       */
72      public boolean checkInternetEnabled();
73
74      public boolean checkWifiEnabled();
75
76      /*
77      * *****
78      *           Base64 Encoding           *
79      * *****
80      */
81      public String encodeTobase64(Bitmap image);
82
83      // File writing
84      public void writeFile(String s);
85
86      @Override
87      public void onSaveInstanceState(Bundle savedInstanceState);
88
89      @Override
90      public void onRestoreInstanceState(Bundle savedInstanceState);
91  }
```


4.2 Android Data Classes

4.2.1 StumblrData {abstract}

Listing 8: StumblrData

```
package uk.ac.aber.cs.groupten.stumblr.data;
2
import android.os.Parcelable;
4
import java.util.Calendar;
6
/**
8  * Abstract class containing basic common structure for all Stumblr data
   formats.
   */
10 public abstract class StumblrData implements Parcelable {
    public static final String TAG = "STUMBLR";
12
    /**
14     * The title of the given piece of data. This can be extended to
       Waypoints, Routes
       * and any other piece of relevant data inside the structure of
       Stumblr.
16     */
    private String title;
18
    /**
20     * The short description pertaining to the given piece of data. This
       will also be
       * extended to other component classes of StumblrData
22     */
    private String shortDesc;
24
    /**
26     * Default constructor
       */
28     public StumblrData() {
        // do nothing
30     }

32     /**
       * @param title The title to set.
       * @param shortDesc The short description to set.
       */
34     public StumblrData(String title, String shortDesc);

36
38     /**
       * Checks the StumblrData item for validity. Returns a boolean. (true
       = valid)
40     *
       * @return Whether the data is valid or not. (true = valid)
42     * <p/>
       * MUST be implemented in any subclasses.
44     */
```

```
    public static boolean isValidData(String s);
46
    /**
47     * Returns current time.
48     *
49     * @return The current time.
50     */
51     public long getCurrentTime();
52
53     /**
54     * Returns the title.
55     *
56     * @return this.title
57     */
58     public String getTitle();
59
60     /**
61     * Sets the current title.
62     *
63     * @param title The title to set.
64     */
65     public void setTitle(String title);
66
67     /**
68     * Returns the short description.
69     *
70     * @return shortDesc
71     */
72     public String getShortDesc();
73
74     /**
75     * Sets the short description.
76     *
77     * @param shortDesc
78     */
79     public void setShortDesc(String shortDesc);
80
81     /**
82     * Sanitises given text by removing prohibited characters.
83     *
84     * @param input The text to sanitise.
85     * @return The sanitised string.
86     */
87     public String sanitiseStringInput(String input);
88 }
}
```

4.2.2 Route

Listing 9: Route

```
1 package uk.ac.aber.cs.groupten.stumblr.data;

3 import android.location.Location;
  import android.os.Parcel;
5 import android.os.Parcelable;
  import android.util.Log;
7
  import java.util.LinkedList;
9 import java.util.Stack;

11 public class Route extends StumblrData implements Parcelable {

13     /**
      * startTime used for timestamp of start of the walk.
15     */
    private long startTime;

17     /**
      * lengthTime used for timestamp for length of walk.
19     */
    private long lengthTime;

21     /**
      * longDesc
23     * A slightly longer description of the contents of the route. Set by
      * the user when
25     * they create a Route.
27     */
    private String longDesc;

29     /**
      * The list of Location objects that reflect the coordinates of the
      * walk.
31     */
    private Stack<Location> coordinates;

33     /**
      * A LinkedList of Waypoint objects that the Route comprises of.
35     */
    private LinkedList<Waypoint> route;

37     /**
      * Constructor. Calls initRoute() to initialise route.
39     */
    public Route();

41     /**
      * Helper method for constructor. Calls initLists() to initialise the
      * list and stack.
43     */
    private void initRoute();

45     /**
      * Helper method for constructor. Calls initLists() to initialise the
      * list and stack.
47     */
    private void initRoute();
```

```
49
    /**
51     * List and stack initializer method.
    */
53     private void initLists();

55     /**
    * Initialise Route object from a Parcel.
57     */
    public Route(Parcel p);

59
    /**
61     * @return The LinkedList of Waypoint objects.
    */
63     public LinkedList<Waypoint> getWaypointList();

65
    /**
    * @return The list of coordinates.
67     */
    public Stack<Location> getCoordinateList();

69
    /**
71     * Adds an item to the list of coordinates.
    * @param location The Location object to add.
73     */
    public void addCoordinate(Location location);

75
    /**
77     * Returns the long description of the Route.
    * @return The long description of the Route.
79     */
    public String getLongDesc();

81
    /**
    * Sets the long description.
83     *
    * @param longDesc A longer description of the Route.
85     */
    public void setLongDesc(String longDesc) {
87         this.longDesc = longDesc;
    }

89

91
    /**
    * Method loops through coordinate list.
93     *
    * If list is empty, return 0.0f. Sets current location by calling
    getLatitude()
95     * and getLongitude() methods. On each loop, increment i and set
    currentLoc to coordinates
    * held in position i of coordinates list. Do same for nextLoc but add
    1 so it retrieves the
97     * coordinates of the position in front of current location. Using
    distanceBetween() it
    * calculates distance between the two locations and add results to
```

```
        distance variable.
99     *
    * @return returns the route distance.
101    */
    public float getDistance();
103
    /**
105     * Describes contents of parcel.
    * @return Description
107     */
    @Override
109    public int describeContents();

111    /**
    * Writes the Route into a Parcel for moving between screens.
113     *
    * @param parcel The parcel to be written to.
115     * @param i      Flags.
    */
117    @Override
    public void writeToParcel(Parcel parcel, int i);
119
    /**
121     * Reads Route data from a parcel.
    *
123     * @param inParcel The parcel to read the Route from.
    */
125    public void readFromParcel(Parcel inParcel);

127    /**
    * Private class that helps create a Parcelable.
129     */
    public static final Parcelable.Creator<Route> CREATOR = new Parcelable
        .Creator<Route>();
131

    /**
133     * @return startTime of the walk
    */
135    public long getStartTime();

137    /**
    * startTime gets set to current time.
139     */
    public void setStartTime();
141

    /**
143     * @param start pass in start time
    */
145    public void setStartTime(long start);

147    /**
    * @return lengthTime variable
149     */
    public long getLengthTime();
```

```
151
    /**
153     * @param l pass in length time and set lengthTime to it.
    */
155     public void setLengthTime(long l);

157     /**
    * @return lengthTime as a casted float divided to be hours.
159     */
    public float getLengthTimeHours();
161 }
```

4.2.3 Waypoint

Listing 10: Waypoint

```
1 package uk.ac.aber.cs.groupten.stumblr.data;

3 import android.graphics.Bitmap;
import android.location.Location;
5 import android.os.Parcel;
import android.os.Parcelable;
7 import android.util.Log;

9 public class Waypoint extends StumblrData implements Parcelable {
    // CONSTRUCTORS
11
    /**
13     * Default constructor for Waypoint.
    */
15     public Waypoint();

17     /**
    * Constructor for a Waypoint object from a Parcel.
19     */
    public Waypoint(Parcel in);

21     // INSTANCE VARIABLES

23     /* title and shortDesc are declared in StumblrData and are accessed
    through get/set methods */
25     /**
    * Arrival timestamp for Waypoint.
27     */
    private long timestamp;

29     /**
    * Image contained within Waypoint.
31     */
    private Bitmap image;

33     /**
    * Location for Waypoint
35     */
    private Location location;

37     /**
    * Helper method for initialising Waypoint objects.
39     */
    private void initWaypoint();

41     /**
    * Sets timestamp.
43     *
    * @param l The timestamp.
45     */
    public void setTimestamp(long l);

47
49
```

```
51      /**
52      * Returns timestamp.
53      *
54      * @return The timestamp.
55      */
56      public long getTimestamp();
57
58      /**
59      * Sets the current location.
60      *
61      * @param l The current location to set.
62      */
63      public void setLocation(Location l);
64
65
66      /**
67      * Sets the current location.
68      *
69      * @param l The current location to set.
70      */
71      public Location getLocation();
72
73      /**
74      * Returns current Bitmap.
75      *
76      * @return The current Bitmap that the Waypoint has,
77      */
78      public Bitmap getImage();
79
80      /**
81      * Sets the current Bitmap.
82      *
83      * @param b The current Bitmap.
84      */
85      public void setImage(Bitmap b);
86
87      /**
88      * Returns a String with the title.
89      *
90      * @return The title string.
91      */
92      public String toString();
93
94      /**
95      * Writes the Waypoint into a Parcel for moving between Activities.
96      *
97      * @param parcel The parcel to be written to.
98      * @param i      Flags.
99      */
100      @Override
101      public void writeToParcel(Parcel parcel, int i);
102
103      /**
```



```
105     * Reads Route data from a parcel.
106     *
107     * @param inParcel
108     */
109     public void readFromParcel(Parcel inParcel);

111     public static final Parcelable.Creator<Waypoint> CREATOR = new
        Parcelable.Creator<Waypoint>();
        public Waypoint createFromParcel(Parcel in);

113
114     /**
115     * @param size pass in size of new array.
116     * @return Waypoint type array with passed in size.
117     */
118     public Waypoint[] newArray(int size);
119 };

121 @Override // Unused
122 /**
123     * Unused method. Required for parcelable implementation.
124     */
125     public int describeContents();
}
```

4.3 Web Interface Descriptions

4.3.1 Index.php

Listing 11: Index.php

```
<?php
2 $walkID = 0;
  if(isset($_POST['tour'])){
4     $walkID = $_POST['tour'];
  }

6
  error_reporting(E_ALL ^ E_NOTICE ^ E_DEPRECATED );
8 $con = mysql_connect ("db.dcs.aber.ac.uk", "username", "password");
  $db = mysql_query("USE csgp10_13_14", $con);
10 $res = mysql_query ("select * from walks where id='$walkID'");
  $query = mysql_query ("SELECT * FROM location WHERE walkID='$walkID'");
12 $query2 = mysql_query ("SELECT * FROM description WHERE walkID='$walkID'")
    ;

14
  $marker[] = 0;
16 ?>

18 //      HTML code that generates webpage
  //      HTML includes the google map functionality
20

<?php
22 while ($a = mysql_fetch_array ($res))
  {
24     echo "<p>" . $a["id"] . "</p>";
    echo "<p>" . $a["title"] . "</p>";
26     echo "<p>" . $a["shortDesc"] . "</p>";
    echo "<p>" . $a["longDesc"] . "</p>";
28  }
  while ($a = mysql_fetch_array ($query)){
30     echo "<p>" . $a["id"] . "</p>";
    echo "<p>" . $a["latitude"] . "</p>";
32     echo "<p>" . $a["longitude"] . "</p>";

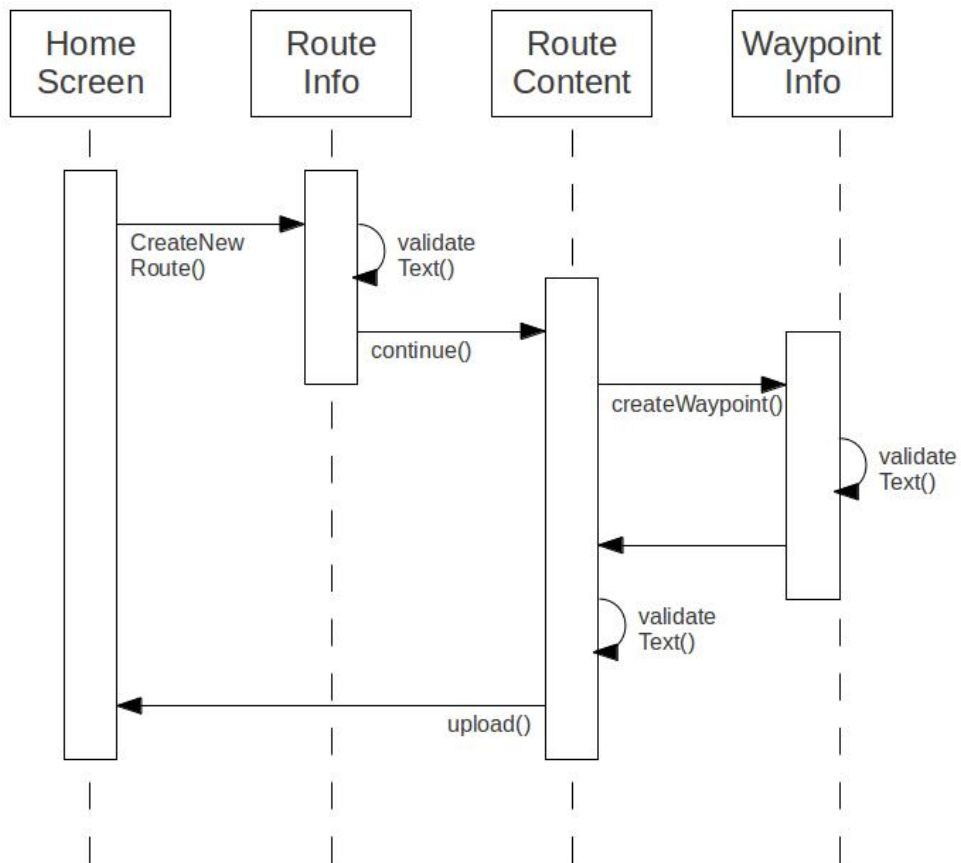
34  }
  while ($a = mysql_fetch_array ($query2)){
36     echo "<p>" . $a["id"] . "</p>";

38     echo "<p>" . $a["description"] . "</p>";
  }
40 ?>

42 //      HTML code that generates table to structure the page
```

5 DETAILED DESIGN

5.1 Sequence Diagram



5.2 Significant Algorithms

5.2.1 Upload Data

1. Bundle data into JSON
2. Send JSON To Server
3. Wait for confirmation...
4. If confirmation timeout reached... prompt user to resend JSON.

5.2.2 Bundle Data

1. Base 64 encode all images
2. Collate data for each waypoint into JSON
3. Collate every waypoint route metadata into JSON

5.2.3 Validate Inputs

1. Check if empty
2. Check if contains "unsanitised" characters
3. Check image sizes

5.2.4 Change Screen

1. Validate Inputs
2. Display next screen

5.3 Significant Data Structures - Android

The main classes in our design are the Route class, and the Waypoint class.

5.3.1 Route

The route class is responsible for storing and processing all the information for one particular route. It does this by storing basic route information, plus a linked list of waypoints. We chose a linked list, as it's size is not fixed, meaning that we are not limited in terms of how many waypoints we can have. We chose a linked list over an arraylist, as an arraylist would be innefficeint and slow, due to the processes involved when adding new items.

5.3.2 Waypoint

The waypoint class is responsible for storing and processing the data for each waypoint along the route. It does this by storing basic route information, plus a linked list of co-ordinates between the current waypoint and the previous waypoint. The timestamp is also generated when the waypoint is constructed. It can also store an optional image, if the user so wishes to add one.

5.3.3 Co-ordinates

The android.location.Location class from the Android API will be used for storing coordinates.

5.4 Significant Data Structures - MySQL Database

To store data about tours a database will be used, this will use MySQL and be the gateway between the android program and the web program. The database shall contain the four tables below with the relevant columns.

5.4.1 List of Walks

Each walk is listed in the "walks" table. There is one record for each walk, and the record has the following characteristics. An ID which is used to order the list of the walks, it will be the primary key for the record and generated by the database. Each walk has a title which represents the name of the walk set by the user and will be of type varchar. The tour also has both short and long descriptions. The short description is a short summary of the walk, where as the long description goes into more detail about the tour. Both of these columns will be varchars. The tour also contains details of how long the walk will take and this will be stored as a float in the database. This will be generated by taking a time for the start of the walk and a time when the walk is finished, then calculating the difference. The final data stored about a walk is the distance and this will also be stored as a float. To create this value the distance between each waypoint will be added.

5.4.2 Location

In this table basic details will be stored about each Waypoint within the walk. The primary key in is the ID and will be generated by the database. This table also uses a foreign key to reference which walk is part of in the walks table, which will be called walkID. Next in the table are two columns for longitude and latitude, both of which are stored as floats. Finally the last column in the table is the timestamp and this will be stored as a float.

5.4.3 Place Description

This table is used to store extra details about each waypoint within a walk. Therefore this table needs an ID as primary key and also a foreign key to link to the location table, which will be called locationID. There is also a column for description which gives a description about the Waypoint. The description will be of type varchar.

5.4.4 Photo Usage

This table is very similar to the place description table, but stores a file path to an image of the location. The table needs to have a primary key (id) for each image as well as a foreign key linked to the location which it is about (placeID). The final column in the table is a varchar which stores the Base64 representation of an image - "photoData".

5.5 Website Interface Specification

The web interface will use all of the following classes to dynamically create a page that will acquire information from a database of routes. The user will also be able to choose from a range of tours provided right next to the map.

5.5.1 Page

The page class is the web page that will hold the map and database interaction. This class will be the way the interface is shown by using css. This class will hold the map class and the database interaction class. The page class will be done in HTML and it will validate to XHTML 1.0 Strict.

5.5.2 Map

The Map class is the class which holds the Google maps API which will be in JavaScript. This class holds functions that are provided from the Google maps API website that initialise the maps and create the pointers on the map. There also other function such as myclick() which records when the user click on the map. Also there is a createMarker() function which will create an info box for each point provided this is were we will provide the information for each place by using the database interaction here to create all the pointers. Below are some functions in JavaScript that the web interface will use with the Google Maps API.

5.5.3 Database Interaction

The database interaction class is where the web page will connect to the databases then It will use JQuery to get all the points on the tour that the user choose. It will present the tours as links to the left of the map which when clicked will connect to that database and load in all the points to the map and load pointer to the right of the map. The map will have information boxes for each point on the map which have had information put in them by using JQuery as the information for each point is save alongside the coordinates. A thumbnail of an image of the location will also be in these information boxes and when clicked on a bigger image will show at the centre of the page. PHP code will be used to read in the databases of tours which will contain each tours the code will create links on the left of the map to bring up a new tour and load in the pointer. The PHP code used will mostly be in the "create pointer" part of the map class where it will create a pointer for each entry in the database which will then create a link on the right of the map to link to each pointer on the map.

5.6 JSON Example

```
{
2   "Walk": {
      "walkTitle": "Demo Walk Title",
4     "shortDescription": "Demo Short Description",
      "longDescription": "Demo Long Description",
6     "walkHours": 42,
      "walkDistance": 42,
8     "Locations": [
        {
10        "LocationTitle",
          "GPSTraces" : [
12          {
            "latitude": 42.00000,
14            "longitude": 42.00000
          }
16        ],
          "timestamp": 16:20,
18          "locationDescription": "Demo Location Description",
          "Photo": {
20            "photoName": "Photo of Location",
            "64bitPhoto": "gfkfhjfhkhfctyb7r67tny8b6n756"
22          }
        }
24      ]
    }
26 }
```

5.7 HTTP Transaction and Expected Values

The JSON transaction is expected to be completed using the HTTP POST protocol. A check for successful upload will be performed by checking the HTTP response code (success = 200).

6 Android Design Justifications

- We decided to implement Base64 encoding on our images within our application - this allows the developers to directly embed the Base64-encoded image data within the webpage - and it's a simple API call to convert to Base64 within Android.
- We also decided to use JSON within the MIME/HTTP POST request. This would allow us to transfer data between client and server in a much more human-readable (and therefore maintainable) manner. Furthermore, JSON is a much more modern standard than mimicking an HTTP form POST, and just as simple to implement. The team spoke to the customer, and they approved the design change.

7 Web Design Justifications

- We decided to use the Aberystwyth Users web server (<http://users.aber.ac.uk/>) for hosting the website. This is because it is fast, reliable, well-maintained and stable, as well as running on a solid Linux platform. The server also runs our chosen services - namely PHP and MySQL. We chose those two technologies as PHP is a long-recognised and popular web programming language, and MySQL has an enormous user base and is extremely stable, as well as implementing all of the SQL functions we plan to use.
- Google Maps was chosen for its simple and well-documented API, as well it's flexible JavaScript library. It also has a very well-known and highly attractive interface, which fits in with our interface design plan.

8 REFERENCES

- [1] GOOGLE. Google Android API Reference.
<http://developer.android.com/reference/android/location/Location.html>.
Accessed: Dec 05, 2013.
- [2] GOOGLE. Google Maps API JavaScript Reference.
<https://developers.google.com/maps/documentation/javascript/3.exp/reference>.
Accessed: Dec 05, 2013.
- [3] HURNHERR, T. L^AT_EX Listings Package Info.
<http://texblog.org/2008/04/02/include-source-code-in-latex-with-listings/>.
Accessed: Dec 03, 2013.
- [4] PAVLIC, T. Programming Assignment template for L^AT_EX.
<http://www.latextemplates.com/template/programming-coding-assignment>.
Accessed: Oct 23, 2013.
- [5] VARIOUS. L^AT_EX Listings Java Highlighting.
<http://tex.stackexchange.com/questions/115467/listings-highlight-java-annotations>.
Accessed: Dec 03, 2013.
- [6] VARIOUS. L^AT_EX Listings JavaScript Highlighting.
<http://tex.stackexchange.com/questions/89574/language-option-supported-in-listings>.
Accessed: Dec 03, 2013.

9 VERSION HISTORY

Author	Date	Version	Change made	CCF
CCN	03/12/2013	1.0	Initial build of document	n/a
DEC	04/12/2013	1.1	Updated Markdown conversion issues	#33
CCN	04/12/2013	1.2	Corrected spelling and grammar mistakes	#34
CCN	05/12/2013	1.3	Inserted missing sections	#34
CCN	05/12/2013	1.4	Rearranged to reflect order in QA doc	#35
CCN	14/02/2014	1.5	Removed broken document markup	#35
CCN	14/02/2014	1.6	Updated interfaces, diagrams and wording	#36

4 REFERENCES

- [1] PAVLIC, T. Programming Assignment template for L^AT_EX.
<http://www.latextemplates.com/template/programming-coding-assignment>.
Accessed: Oct 23, 2013.
- [2] TIDDEMAN, B. P. Software Engineering Group Projects - Quality Assurance Documents.

5 VERSION HISTORY

Author	Date	Version	Change made
CCN	07/11/2013	1.0	Updated order of authors on cover