CS673F14 Software Engineering
Group Project - Group 5 - eDocent (mobile museum guide)
Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Casey O'Rourke	Leader		9/25/14
Kritika Balasubramanian	Testing Leader		9/25/14
Chethan Kolige Chandrappa Gowda	Integration Leader		9/25/14
Nisreen Dahod	Design Leader		9/25/14
Sukaynah AlMutawa	Implementation Leader		9/25/14
Ashley Custer	Configuration leader		9/25/14

Revision history

Version	Author	Date	Change

1. Overview

Surveys have shown that museums are “enthusiastic about the potential of mobile technology” and an important part to any visitor engagement strategy[1]. However, many institutions are faced with both financial and technical constraints preventing the development of such tools for mobile users. While larger institutions are able to afford custom applications, many institutions do not possess the resources to have their own application built and less than half of surveyed museums offer mobile experiences for visitors[2].

Our group seeks to develop an open source solution for museums to engage their users through mobile applications. The effort is two fold: museums need the necessary tools to upload and organize content, while the visitors require a mobile application to access the content. The first tool will consist of a web interface for museums to upload content and will also implement a web service for the data to be pulled from. The mobile application will be able to select from a list of museums and take users to the museum’s main front page. From there, users of the mobile application can lookup relevant information about the museum, ranging from the museums hours to looking up extra information about a painting via QR scanner.

The potential users would not be limited only to museums, but could potentially be used for exhibitions, fairs, gallery showings and other events where people may need to have access to extra information to enhance their experience of the event. The mobile application will serve as a portal to any content uploaded by these institutions. For a later iterations, users may be able to sign up for accounts and provide the mobile application with a social media experience as well (or to integrate with Facebook or Twitter). This would allow users to leave comments, save favorite objects and share their experiences with others. It would also allow the museum to have a better idea of who their visitors are.

[1] http://www.aam-us.org/docs/center-for-the-future-of-museums/aam_mobile_technology_survey.pdf?sfvrsn=0
[2] *ibid*

2. Related Work

As mentioned, some museums have created their own customs applications. One of the best examples of this is the Museum of Modern Art's mobile application (iOS and Android). The application provides a portal for users to examine the collection, see which items are on view, look up terms in an art glossary, listen to audio guides, and many other features. The Smithsonian institution also has a mobile application which serves as portal to the many museums that it consists of. This aspect of being a single application to that points to many other institutions is very similar to what we hope to achieve.

We hope to implement the same set of functionality for users of the mobile application, but to ultimately have the mobile application be usable by any institution. Users will select their institution of interest upon startup of the application to access relevant content. By allowing museums to upload content, they can then populate the application with their information, which can then be used with the mobile application to enhance the museum going experience. This will allow museums to skip the expense of developing a custom application and instead focus on content and visitor engagement.

3. Detailed Description

The project will consist of two components: 1) the web application and service for uploading, hosting and transmitting content and 2) the Android mobile application for accessing the content. The user of the web interface will be the museum/institution, while the user of the mobile application will be the visitors to the museum/institution.

The web application will use the Django framework, an open source web application framework written in Python. It will allow the user to create entries for their collection, events and general information about the institution. Content related to the collection can be textual, audio or visual. In uploading an object, each item will be assigned a unique code and the museum can display this code either as a number or QR barcode. They can then place this code with the objects description, allowing the user to lookup or scan the code for more content.

The mobile application will open onto a screen where the user will select/search for the institution they are visiting. Once found, they will be brought the museums main page which will display relevant information about the museum (hours, map, ticket prices). Other menu items will include the ability to search the collection, lookup items or select an audio tour. As our project progresses, we hope to ultimately add more features.

Web Application and Service

- Functional
    - Secure Museum login
    - Upload/edit/delete content
        - Text content
        - Audio/visual content
        - Guided tours
        - Museum Map
    - Post museum information (hours, map, ticket prices, etc)
    - Allow for content to be pulled from mobile application
- Non-functional
    - Django framework
    - Use REST protocol to talk with mobile application
    - PostGreSQL database
    - Two-factor authentication login
    - Hosted on either Amazon Web Services, Heroku or Google App Engine
    - Separate media server on AWS for audio and visual content

- Future features:
    - Batch upload of objects from database
    - View info about mobile users
    - View statistical information about objects
    - Ask a curator function

Mobile Application

- Functional
    - Allow user to select institution from ordered list or search
    - Display main page about museum
    - Search selected museum's collection
    - Direct look up of item through QR barcode or unique code
- Non-functional
    - Android application
    - Uses REST protocol to talk to web service

4. Management Plan

a. Process Model

In our project, we will be using Agile approach, which is an iterative and incremental development process. The total time restriction is about ten weeks; thus, every three weeks will present one complete iteration. Each iteration will produce the product with additional features than the previous iteration, including planning, requirement analysis, design, coding, and testing. The main purpose of these iterations is to inform the customers with the most recent development, so we make sure that our project is following the customers' desired; they can interact to us, and possibly do slight change to the requirement.

We will also be utilizing a team communication tool to help implement a loose version of the scrum framework. With two teams, each team is able to work independently towards its own goals. Using a communication tool service called "Slack", members have been and continue to communicate on a daily basis over progress and what needs to be done. It also allows members of different teams to observe to keep track of each others progress.

b. Objectives and Priorities

The objective is to deliver a high quality and thoroughly tested product that we hope can ultimately be put to use by museums. Given the customer, we hope to develop a product that is intuitive to use. Within the timeframe of this course, we will focus on working prototypes over documentation. We hope to remain agile, working together in small teams that can react quickly to change.

c. Risk Management

1. Risk with respect to work to be done:
    a. Miscommunication: After a meeting, a group member should summarize the discussion in a report, and each member should have access to the report. Team members should not hesitate to ask or re-ask questions if things are unclear. If miscommunication is causing any problem, another meeting should be organized to clear things up.
    b. Time Shortage: Care should be taken to plan enough spare time. Tasks could be completed before time or could fail to finish on time, by providing some buffer time project planning can be adjusted. Also, if time shortage becomes severe, user requirements could be dropped after consultation based in priority.
    c. Design Errors: The design should be reviewed very critically. The leaders in each group should be consulted frequently on his opinion about the feasibility and the correctness of certain design decisions.

All the work that depends on the faulty design should be halted until corrected.

d. Server Crash: All products should be stored in the project repository, which should be backed up regularly by the leaders. When a product gets lost from its working store, it is recovered from the most recent backup.
e. Lack of Testing: It is very easy for the project to get ahead of its testing. While coding new features is exciting and rewarding, care should take to ensure the code is thoroughly tested before adding more features. Failure to do so with will almost certainly result in delays.
f. Technology: Some members of the group are more familiar with the technologies and others. As a result, it is critical the group ensure that everyone is on the same page as to what is going on in terms of technical understanding.

2. Risk with respect to team members/Project Manager:
    a. Team members should warn their team leader or Project Manager timely before a planned period of absence. By ensuring knowledge is shared between team members, work can be taken over quickly by someone else if a person is not present. When works needs to be taken over, a re-division is made so that the workload is not too high.
    b. The project manager should inform the assistant project manager of absence in time so that she can prepare to take over. By keeping assistant project manager up-to-date on the project status, she will have enough knowledge to take over.

Summary:

1. Try to signal problems as early as possible and report them to the Project Manager, so that action can be taken
2. Pay attention to communication and make sure everybody understands the things the same way
3. Focus on the agreed user requirements, which express the wishes of the customer
4. Minimize the friction between people by helping and supporting each other
5. Coordinate and follow other guidelines (for example coding) to ensure product quality

d. Monitoring and Controlling Mechanism

i. Monitoring Mechanism: The work being performed must diligently be monitored and tracked. This must be done to ensure the correct activities

are being done, the team is on schedule, the team's work quality is acceptable, and to point out when the project is deviating from what the customer wants. To do this, progress must be scheduled and each feature should be in scope of the project. During each weekly meeting, each member will state what they will be working on for that week and the status of each feature they are implementing.[3] In addition, "Slack" will be used to update the team on progress and send users notifications of any changes to the GitHub repository.

ii. Controlling Mechanism: Controlling Mechanism ensures that the project is delivered on schedule. Besides progress reports during each meeting. If any features are deemed unneeded and time is less, as a group, we should reconsider whether it is essential to have or not. If it is essential, then we should reevaluate what may not be implemented to account for the extra time needed for this essential feature.

e. Schedule and deadlines

Iteration 0 Initial Planning Phase

Deliverables due end of week 3, September 25, 2014:

1. proposal, SPMP
2. presentation

Schedule:

Week	Dates	Plan	Deliverables
3	9/19 - 9/25	● Monday meeting ● Individual learning	● meeting minutes ● weekly report ● 3 user stories about our project and 1 user story as customer for other team's project (per person, written on pivotaltracker) ● get comfortable with github, communication tools, and programming tools ● assign responsibilities ● presentation ● SPPP document

Iteration 1

Deliverables due end of week 6, October 16, 2014:

[3] http://www.theprojectmanagementsteps.com/project-management-steps/monitoring-controlling

1. presentation and demo
2. all source code and test cases
3. SDD and updated SPMP

Code goal: Museums will be able to add all their information including address, hours, tour times, and special exhibits. A REST protocol will communicating with Android application, and the Android interface will provide a visual of museum information.

Schedule:

Week	Dates	Plan	Deliverables
4	9/26 - 10/2	Monday meeting	● meeting minutes ● weekly report
5	10/3 - 10/9	Monday meeting	● meeting minutes ● weekly report
6	10/10 - 10/16	Monday meeting	● presentation and demo ● all source code and test cases ● SDD and SPMP

Iteration 2

Deliverables due end of week 9, November 6, 2014:

1. presentation and demo
2. all source code and test cases
3. SDD and updated SPMP

Code goal: Museums will be able to upload art object, and the Android application will have access to these objects through search or browse.

Schedule:

Week	Dates	Plan	Deliverables
7	10/17 - 10/23	Monday meeting	● meeting minutes ● weekly report
8	10/24 - 10/30	Monday meeting	● meeting minutes ● weekly report
9	10/31 - 11/6	Monday meeting	● presentation and demo

			● all source code and test cases ● SDD and SPMP

Iteration 3

Deliverables due end of week 12, December 4, 2014:

1. final presentation and demo
2. final source code and test cases
3. SDD and updated SPMP

Code goal: Museum patrons will be able to scan QR code to directly look up pieces of art.

Schedule:

Week	Dates	Plan	Deliverables
10	11/7 - 11/13	Monday meeting	● meeting minutes ● weekly report
11	11/14 - 11/20	Monday meeting	● meeting minutes ● weekly report
12	11/21 - 11/27	Monday meeting	No class, Thanksgiving
13	11/28 - 12/4	Monday meeting	● final presentation and demo ● final source code and test cases ● SDD and SPMP

5. Quality Assurance Plan

a. Metrics

i. Fault Profile: This Metric would be used to log errors in modules. We would then need to prioritize and categorize errors. Errors would be logged in a shared document on Google Documents. We would hope the number of errors resolved is high or 100% to analyze the metrics for process improvement.

ii. Breadth of Testing: This Metric provides an indication of completeness of testing. We would need to test every module and functionality in the front

and back end. We will keep a log shared document, to keep track of the number of modules/functionality created and if the testing is complete at each iteration to analyze the process improvement.

b. Standards

i. Documentation Standards

1. Source code for the latest revision of a document must be available on GitHub for the rest of the group to access.
2. Documentation of code must be placed in a separate documentation as well as comments within the code.

ii. Coding Standards

1. Each class, method, and function will be identified by their own comment header. For a class, the header must include class name, author, and purpose of the class. The header for methods and functions should include a description, purpose, and if there are any, then also parameter arguments and the return value.
2. Naming conventions must be adhered to with classes/objects, methods/functions, and variables. Class and object names should be nouns, capitalized, simple, and descriptive. They should not start with an underscore. methods and functions should be verbs and the first letter of the method/function name should not be capitalized. Finally, variables names should not be reused, except in trivial cases like loops and indices. Class names, object names, method names, function names, and nontrivial variable names should be at least two characters long.
3. Using Charles SImonyi's Hungarian Naming Convention.To come up with a list of standard prefixes for the project and to make developers use a consistent standard in naming variables.
4. Formatting the code such that it helps the developers to understand the basic structure of the code with a glance.Using consistent indentation, using upper case letter with underscore to express constants, aligning the open and close braces are some of the formatting standards.

iii. Comment Standards

1. Like stated above, each class, method, and function must be identified by their own comment header using the "JavaDoc" notation convention. Each comment header must include the purpose of the class, function, and method. Each comment header must also include what the function and method may return and any parameters the class, method, and function may have. Methods and

functions should also have many in-line comments, which describe what is occurring within the method/function.

2. Java Comment Standard: For Java, documentation, or block quotes, must start with “/**” and end with a “*/”. A one-line comment begins with a “//”. One-line comments must be used to describe implementation details, such as the purpose of variables and constants.
3. Python Comment Standards: For Python, “#” precedes in-line comments. In-line comments are used within methods and functions and describe implementation statements. They are on the same line as statements and are at least two spaces away from the statement. For Python, header comments must be implemented using documentation strings (docstrings), which sandwich the comments with a “ ” ”.
4. Software Documentation can be of two forms a) External and b) Internal.Those which are outside the source code files come under external documentation. Internal documents are those which are written by the developers when creating new or altering the existing code. Each modification to the code should be documented internally. However, these comments do not affect anything at runtime. But they are very important and immensely time saving when it come to maintaining intricate pieces of software.

c. Inspection/Review Process

All code is subject to review when a bug or defect is found. If it is code for the web interface, then the web interface group should conduct the review, with the author of the code. If it is code for the mobile application, then the mobile application group will conduct the review. An inspection meeting will be held, where the author will read the code and the other members will point out any bugs or defects. The author of the code will then fix the code according to the actions of the inspection meeting. Then, the author will make sure what was changed is correct. Each group’s leader will be the moderator. The author of the code will also be the reader. The leader and the third group member will be the inspectors. Finally, the third group member will also be the scribe, who writes records any defects found.

d. Testing

To guarantee that our project has satisfied the specified requirements, testing must take place. The STP (or Software Testing Plan) will be formed in the second phase of the project development. The result of testing shall be recorded in order to keep track of the project’s overall performance. Testing will be applied in the three following ways:

- *Unit Testing:* This type of testing must be done frequently after each

small modification. It will be done by the programmer who wrote that code himself.
    - Front-end:
        - For the android front end testing, we would run each test case class in Eclipse with ADT, the output appears in the Eclipse JUnit view. We can debug and fix errors if any.
    - Back-end:
        - Testing the models files, specifically the classes that represent the tables of the database.
- *System Testing:*
    - Front-end:
        - Testing the android app with all its part as a whole to assure that this “half” of the project works. This shall be done by the front-end part leader: Chethan.
    - Back-end:
        - Testing the whole Django and Django-REST framework, seeing whether the entire “half-project” works or not. This shall be done by the back-end leader: Casey.
- *Integration Testing*: This type of testing shall be done once at the end of each iteration by the quality assurance leader and the quality control team.
        - Testing the entire project as one, including front-end and back-end, making sure that the different parts of the system work together successfully .

e. Defect Management

i. We define a defect to be any flaw or imperfection in a software work product or a software process. A software work product is any artifact created as part of the software process including computer programs, plan procedures, and associated documentation and data.

ii. We categorize the defects in the following severity level:
    1. Critical - When the defect could affect the entire project.
    2. High - The bug or issue affects a crucial part of the project, and must be fixed in order for it to resume normal operation.
    3. Medium - The bug or issue affects a minor part of the project, but has some impact on its operation.
    4. Low - The bug or issue affects a minor part of the project, and has very little impact on its operation (and with lower importance).

5. Cosmetic - The system works correctly, but the appearance does not match the expected one. For example: wrong colors, too much or too little spacing between contents, incorrect font sizes, typos, etc. This is the lowest priority issue.

iii. To categorize the defects based on priority, we would rate the defect on 1 to 5, 1 being the highest priority and 5 the lowest.

iv. We will use Pivotal Tracker to handle defects. We will create a User Story in Icebox and select the Story Type as Bugs,we will assign the task to the respective team member. We will push the task in current/icebox/backlog as required.

6. Configuration Management Plan

a. Configuration items and tools

GitHub will be used as the project's version control tool. Frontend code created in Eclipse will be committed using a GitHub plugin. Communication about version control will take place on the team's Slack page.

b. Change management and branch management

Initially, the GitHub project will have two branches, one for Android and frontend code and one for Django database management and backend code. In testing novel features and experimenting with techniques, new branches will be created off of the original two. Branches will be merged with the main front end or back end branch only after code is functional and tested. Therefore, the main two branches should always have working code. Before merging feature branches with frontend or backend main branches, the team will be notified of the merge to prevent major conflicts.

c. Code commit guidelines

Follow the rule of commit early and often. However, untested work will be done on isolated branches. Code that can be generated from existing code will not be committed. This includes compiled files such as file.pyc are Java jar files. Eclipse users will not commit meta data files, as these may disrupt work on other machines.

7. References

- "Django Documentation | Django Documentation | Django." *Django Documentation | Django Documentation | Django*. N.p., n.d. Web. 25 Sept. 2014.
- "Introduction to Android." *Android Developers*. N.p., n.d. Web. 25 Sept. 2014.
- "Mobile Technology Trends." *Center for the Future of Museums -*. N.p., n.d. Web.

25 Sept. 2014.
- http://wwwis.win.tue.nl/2M390/projects/spingrid/spmp.pdf
- http://wwwis.win.tue.nl/2M390/projects/spingrid/sqap.pdf

8. Glossary