Prof-desk Senior Project

Winter Quarter Requirements Document

Project Member	
Kyle Bippus	bippuskw@rose-hulman.edu
Signature	Date

Project Member	
Andy Chen	chena1@rose-hulman.edu
Signature	Date

Project Member	
Jeremy Tramm	trammjn@rose-hulman.edu
Signature	Date

Project Member	
Demetruis Vassar	vassardm@rose-hulman.edu
Signature	Date

Table of Contents

REVISION HISTORY

Version	Date	Comments
1.0	12/5/2013	Created initial skeleton and transfer of previous project plan content. (DV)
1.0.1	12/7/2013	Added sections Design Constraints and User Environment (DV)
1.1	12/7/2013	Removal of the sections: key user/stakeholder needs, product perspective, solution constraints, hardware and software interfaces, domain and data modeling, system sequence diagram, coding standard, architecture diagram (DV)
1.1.1	12/7/2013	Added User/Stakeholder section (DV)
1.1.2	12/7/2013	Added Change Control section (DV)
1.1.3	12/7/2013	Added Non-Functional Requirements (DV)
1.2	12/8/2013	Integrated Andy's and Kyle's current Feature Sections (DV)
1.2.1	12/8/2013	Added Andy's Use Cases for his Features (DV)
1.3	12/10/2013	Removed the Current Background sections and added the Current System, Alternatives and Competition, and Documentation and Installation Requirements (DV)
1.3.1	12/10/2013	Added Kyle's remaining features (DV)
1.3.2	12/10/2013	Begin applying Andy's Use Cases (DV)
1.4	12/11/2013	Begin applying Kyle's Use Cases (DV)

INTRODUCTION

This document will serve as the project management plan for the Prof-desk Senior Project.

Prof-desk is an online course-management website for students and professors. Students can collaborate on homework, answer questions, see grades, and upload documents. Professors can upload course materials, grade documents, and create other custom content for their students. Prof-desk would like to have a mobile application as an interface to the main service. From this application students will have the ability to perform many of the same functions as the website (such as viewing grades or uploading documents).

In this document we will outline the key features that will be necessary to deliver a feature-complete product to Prof-desk. We will also discuss software lifecycle plans, risk factors and management, quality attributes, and task estimation. This is a living document and is subject to change in the future.

PROBLEM STATEMENT

Prof-desk believes that education is for everyone. By decreasing barriers to and increasing transparency and coherency good education and by providing the tools students need to succeed, students of all ages, skill level, and languages can gain a better understanding of the subjects at hand and what they need to learn about them. We believe that a more complete educational opportunity and education profile can also help to change the way the world develops ideas, works on projects, hires employees and provides opportunity for everyone. Simple tools that make the learning process easier and more accessible will begin to change the way that people learn and the way that people interact with education. The RHIT Senior Design Group, 2013 will be developing a mobile application that will start to make the communication within the education environment easier!

CURRENT SYSTEM

Prof-Desk's current system is a Learning Management System written in PHP that, currently, only handles traffic from web browsers. In this system, students are able to complete actions that are common to most Learning Management Systems such as getting messages and content from their professors, turning in assignments, and checking their course's calendar. Prof-desk allows professors to create new classes, add students to classes, and add new content, assignments, and quizzes to their classes. However, Prof-desk tries to make learning a more social experience and allows students and professors to add friends and message their friends or writing in their chat to get assistance on class work.

Prof-desk currently has no application available on mobile devices.

ALTERNATIVES AND COMPETITION

Prof-desk's main competition is other Learning Management Systems like Moodle and ANGEL. While these two systems are hosted in more colleges than Prof-desk presently, their existing mobile applications have many problems and are very buggy. Many users have problems doing simple things like logging in and give up trying to use the apps simply because they do not find them useable. A working, feature-filled mobile from a competitor would be useful to stand out from Moodle and ANGEL.

User/Stakeholder Profiles

Student

Profile Type	Description
Representative	A user with the “Student” status. This user type can be any given person who wishes to learn something new academically with the system. With that focus in mind, the given persons are most likely to be students varying the ages of: • Middle School • High School • College
Description	Students are users who are assigned to courses given by instructors through Prof-desk. They have access to course material, access to students from a given course, able to communicate with other students enrolled in a course, and utilize the Prof-desk study material to benefit their learning objectives.
Type	It is assumed that the user is a “casual user” with no technical knowledge of the system.
Responsibilities	Student accounts are presumed to follow the following: • Enroll in courses • Take quizzes and tests from instructors • Communicate with other students • Ask instructors for assistance • Access to course material
Success Criteria	It is assumed that the user would be successful with the system if the system behaves correctly.
Involvement	See Responsibilities.
Deliverables	Student accounts have the ability to turn in homework assignments to the server through images.
Comments/Issues	None at this time.

Teacher

Profile Type	Description
Representative	A user with the “Teacher” status. This user type can be any given person who wishes to teach something new academically with the system. With that focus in mind, the given persons are most likely to be students varying the ages of: • Teaching/Graduate Assistants • Teachers (all grades) • Professors (all levels)
Description	Teachers are users who created courses given by through Prof-desk. They have access to create course material, grades students in a given course, able to communicate with other students enrolled in a course, and utilize the Prof-desk study material to fulfill their learning objectives.
Type	It is assumed that the user is a “casual user” with no technical knowledge of the system.
Responsibilities	Teacher accounts are presumed to follow the following: • Create courses • Add and modify course material • Grade quizzes and exams • Access to course material • Update students about the course

Success Criteria	It is assumed that the user would be successful with the system if the system behaves correctly.
Involvement	See Responsibilities.
Deliverables	See Responsibilities. This will be important to Student account interaction.
Comments/Issues	None at this time.

Feature Listing

Features

Feature Number	Feature Name
1	Login and Authentication
2	Change Profile
3	Send Message
4	Access Course Materials
5	Turn In Assignments
6	Test and Quiz Creation
7	Take Tests and Quizzes
8	Check Grades
9	Ask for Help
10	Access Calendar
11	Calendar Reminders
12	Calendar Group Invites
13	Posting to Facebook
14	Posting to Twitter
15	Access to Test and Quiz Material
16	Flash Cards

Attributes

Our team will use the following feature attributes to evaluate, track, prioritize, and manage the product items proposed for implementation. These attributes will be used to:

Attribute Property	Description
Status	Tracks the progress during the definition of the project baseline (i.e. what is proposed, what will be implemented, etc.) This attribute can take the “Proposed”, “Approved”, and “Incorporated” values.
Owner	Tracks which team member was responsible for creating the write up about the feature. It will help team member and clients clear any inconsistency or confusion.
Priority	Tracks the priority of implementation of the features. It will allow team members to decide which features are absolutely necessary to satisfy the client’s needs and which ones can be implemented later based on time constraints. This attribute can take the “Critical”, “Important”, and “Useful” values.
Effort	Measures the work amount that the team has to put in to implement each feature. It will be used to determine if it is reasonable to implement the feature or not. The exception to this is the “critical priority” feature, which have to be implemented regardless of the amount of effort required. This attribute can take the “High”, “Medium”, and “Low” values.
Risk	Measures the risk of experiencing undesirable events, such as schedule delays. It will also be used, if it is feasible to implement the feature or not. Caution needs to be taken with features that are critical and have a high risk rating. This attribute can take the “High”, “Medium”, and “Low” values.
Stability	Measures the probability of the change in how the team understands the feature. It will help establish development priorities and determine which features have to be fleshed out more than the others. This attribute can take the “High”, “Medium”, and “Low” values.
Assigned To	Tracks those responsible for implementing the feature.
Notes/Reasoning	Records optional information about a feature if the team believes that an explanation is required.

ID	Feature	Status	Owner	Priority	Effort	Risk	Stability
1	Login and Authentication	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: The login view needs to have inputs that take an input for the user’s email and password. A login button will be also present in the view to indicate where the user can continue.			If the email/password combination, the user will be brought back to the view and prompted again for the correct credentials. If the user is successfully authenticated, a session token will be stored on the phone for future logins. This token is also stored on the server to prevent spoofed sessions. *Users should be able to login and authenticate with the existing Prof-Desk system on the Mobile Application.*		The login view needs to have inputs that take an input for the user’s email and password. A login button will be also present in the view to indicate where the user can continue. If the email/password combination, the user will be brought back to the view and prompted again for the correct credentials. If the user is successfully authenticated, a session token will be stored on the phone for future logins. This token is also stored on the server to prevent spoofed sessions.	
2	Change Profile	Proposed	KB	Critical	Medium	Low	Medium
	Notes/Reasoning: Interpreted as “log in as another student/teacher.” A user should be about to log out like he or she could on the website and a new user could log in.						
3	Send Message	Proposed	KB	Critical	Medium	High	Low
	Notes/Reasoning: User can send a message to another user from his device, like from the website.					Client-side processing won’t be difficult. The biggest risk I think is familiarity with how the server handles this function.	
4	Access Course Materials	Proposed	KB	Critical	Medium	Medium	Medium
	Notes/Reasoning:						
5	Turn In Assignments	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: This particular view will allow a student to upload their homework assignments that they are assigned via their phone by uploading a file or other means into a designated dropbox.			*Students should be able to turn in homework assignments via the Mobile Application.*			
6	Test and Quiz Creation	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: This particular view will allow instructors			*Instructors should be able to create*			

	to create tests/quizzes with any number of questions of multiple answer types (e.g. multiple choice or short answer). Instructors will be able to open/close the quiz/test.			*tests/quizzes via the Mobile Application.* This particular view will allow instructors to create tests/quizzes with any number of questions of multiple answer types (e.g. multiple choice or short answer). Instructors will be able to open/close the quiz/test.			
7	Take Tests and Quizzes	Proposed	KB	Useful	High	High	Low
	Notes/Reasoning:					The high risk, high effort of this feature means that we may not have enough time to implement given our timeframe.	
8	Check Grades	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: Students will be able to navigate to a view that displays an overview of all the grades that he/she is currently earning in active classes. Upon selection of a specific class, the student is taken to a more detailed breakdown on his/her current grade in the selected class.			*Students should be able to check their grades in a class.* Students will be able to navigate to a view that displays an overview of all the grades that he/she is currently earning in active classes. Upon selection of a specific class, the student is taken to a more detailed breakdown on his/her current grade in the selected class.			
9	Ask for Help	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: A view will exist where any user can submit a question on any particular subject matter. Users will have the opportunity to choose the main/sub-topics and the subject. There will be input fields for the question title and further content.			*Users should be able to ask for help.* A view will exist where any user can submit a question on any particular subject matter. Users will have the opportunity to choose the main/sub-topics and the subject. There will be input fields for the question title and further content.			
10	Access Calendar	Proposed	KB	Critical	Medium	Medium	Medium
	Notes/Reasoning:						
11	Calendar Reminders	Proposed	KB	Critical	Medium	Medium	Medium
	Notes/Reasoning:						
12	Calendar Group Invites	Proposed	AC	Critical	Medium	Medium	Medium

	Notes/Reasoning: A view will exist where any user can invite other members into a group in the calendar. The view will prompt the user for members to invite and the group to invite the invited members to.			*Users should be able to invite others to their calendar group.* A view will exist where any user can invite other members into a group in the calendar. The view will prompt the user for members to invite and the group to invite the invited members to.			
13	Posting to Facebook	Proposed	DV	Useful	Medium	Medium	Medium
	Notes/Reasoning: Client desires an option for a student or teacher to send updates to their personal Facebook account relevant to the course he or she is taking.			The Facebook posts will just use some type of metadata as its content. It will be convenient for the user to use and post to update students.		System needs to be able to recognize the content and find a stable way to send to Facebook properly.	
14	Posting to Twitter	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: A button/view will exist to post/share the current page on Twitter.						
15	Access to Test and Quiz Material	Proposed	DV	Critical	High	Medium	Medium
	Notes/Reasoning:						
16	Flash Cards	Proposed	AC	Critical	Medium	Medium	Medium
	Notes/Reasoning: A view will where any user can create/edit and use flash cards to aid with memorization						

Use Cases

Use Cases to Feature Mapping

Use Case 1 – Login and Authentication

Use Case Property	Description
Actor(s):	• User • The logic Application Programming Interface (API)
Pre-Condition:	User is not logged into the system
Post-Condition:	1. User remains unauthenticated 2. User is authenticated and has a session key containing his/her session identification string
Flow:	Normal Flow 1. System prompts the user to log in; see figure 4.1 2. User enters his/her username 3. User enters his/her password 4. User confirms submission 5. App attempts to authenticate the user's credentials using the login API 6. App successfully authenticates the user 7. App redirects to app dashboard (or something similar) Alternate Flow(s): 2. a. User does not know his/her username. User is unable to log in 3. a. User does not know his/her password. User is unable to log in 4. a. Server fails to receive the form data. App notifies user about not being able to login at this time. 5. a. Credential authentication fails. App prompts the user to log in again, alerting him/her that the credentials were incorrect. Flow resets to state 2.

Use Case 2 – Change Profile

Use Case Property	Description
Actor(s):	• User • The login Application Programming Interface (API)
Pre-Condition:	User is logged into the system
Post-Condition:	1. The user is at the home screen in a new profile. 2. The app is in the same state as before a log out attempt was made.
Flow:	Normal Flow: 1. User taps the Settings icon in the Action Bar. 2. User taps the option "Log out". 3. App opens a dialog confirming that the user logs out. 4. App uses the login API to log out the user. 5. App redirects to the log in screen. 6. User logs in with new credentials (see Use Case 1) Alternate Flow 1 – User cancels the process: 4. User taps "Cancel" at the confirmation dialog. App returns to its previous state. Alternate Flow 2 – User loses connection:

	5. Connection is lost while the app is attempting to log out. App notifies the user that it failed to log out. Alternate Flow 3 – No internet connection: 5. There is no connection. App notifies the user that it failed to log out.

Use Case 3 – Send Message

Use Case Property	Description
Actor(s):	• User • API for Sending Messages
Pre-Condition:	User is logged into the system User is viewing list of friends
Post-Condition:	• User is returned to the friends list and the message is sent. • User is returned to the friends list and the message is not sent.
Flow:	Normal Flow: 1. User taps on a friend in the list. 2. User taps the option to send a message 3. User types in a subject in the "Subject" text field 4. User types in a message in the "Message" text field 5. User taps the "Send" icon 6. App notifies the user that the message was sent Alternate Flow 1 – No internet connection: 6. App notifies the user that there is no connection, and the message is not sent. Alternate Flow 2 – User loses connection: 6. App notifies the user that an error occurred while sending the message, and the message is not sent. Alternate Flow 3 – "Message" body is empty: 6. App notifies the user that the message body is empty and it must not be empty first. Alternate Flow 4 – "Subject" line is empty: 6. App notifies the user that the subject line is empty and gives the user the option to send anyway or cancel sending Alternate Flow 5 – User cancels sending message: 1. User taps the "Cancel" button 2. App redirects user back to friends list

Use Case 4 – Access Course Materials

Use Case Property	Description
Actor(s):	**TBA**
Pre-Condition:	**TBA**
Post-Condition:	**TBA**
Flow:	**TBA**

Use Case 5 – Turn In Assignments

Use Case Property	Description
Actor(s):	• Student • Homework Turn-In Application Programming Interface (API)
Pre-Condition:	Student is authenticated and navigates to a dropbox.
Post-Condition:	1. Homework assignment is submitted. 2. Homework assignment is not submitted.
Flow:	Normal Flow: 1. Student selects input button to specify a file to upload 2. Student confirms form submission 3. App posts meta-data and specified file through the API. 4. App prompts user that the assignment has been submitted. Alternate Flow(s): 2. a. Student does not submit a file. An error pops up to notify the student to specify a file for upload. 3. a. Server fails to receive the form data. App notifies the Student to try again. Flow resets to state 2.

Use Case 6 – Test and Quiz Creation

Use Case Property	Description
Actor(s):	• Instructor • Test/Quizzes API
Pre-Condition:	Instructor is authenticated and indicates to create a test or quiz.
Post-Condition:	1. Test/Quiz created. 2. Test/Quiz is not created.
Flow:	Normal Flow 1. Instructor indicates name for test/quiz. 2. Instructor confirms name and indicates to continue. Current state is saved. 3. Instructor creates a question and fills in question content. 4. Instructor confirms question creation and indicates to continue. Current state is saved. 5. Instructor creates possible answer(s) and fills in answer content. 6. Instructor confirms answer(s) creation and indicates to continue. Current state is saved. 7. Flow resets to state 2 if instructor has more questions to create. If instructor is done, instructor should indicate to save the test or quiz. Alternate Flow(s): 2. a. Current state cannot be saved due to server error. App notifies instructor to try again at a later time. 3. a. No question content entered. Instructor prompted to enter question content. 4. a. Current state cannot be saved due to server error. App notifies instructor to try again at a later time. 5. a. No options specified or content entered for answers. Instructor is prompted to fill in details for answers. 6. a. Current state cannot be saved due to server error. App notifies instructor to try again at a later time. 7.a. Test/Quiz cannot be saved due to server error. App notifies instructor to try again at a later time.

Use Case 7 – Take Tests and Quizzes

Use Case Property	Description
Actor(s):	**TBA**
Pre-Condition:	**TBA**
Post-Condition:	**TBA**
Flow:	**TBA**

Use Case 8 – Check Grades

Use Case Property	Description
Actor(s):	• Students • Grades API
Pre-Condition:	Student is logged into the system and is on the grades overview view
Post-Condition:	1. User views grades 2. User does not view any grades
Flow:	<u>Normal Flow:</u> 1. User indicates to view grades. <u>Alternate Flow(s):</u> 1.a. App fails to retrieve data from Grades API. Student is prompted with error. 1.b. No grades available. Student is prompted with this message.

Use Case 9 – Ask for Help

Use Case Property	Description
Actor(s):	• Users • Help/Q&A API
Pre-Condition:	User is logged into the system and is on the main view of the app.
Post-Condition:	1. A question is submitted 2. A question is not submitted
Flow:	<u>Normal Flow:</u> 1. User indicates to “Ask For Help/Ask A Question” 2. User fills in main/sub-topics, subject, and content input fields. 3. User confirms form submission and submission is posted to the API. 4. User is redirected to the respectively created Q&A view. <u>Alternate Flow(s):</u> 2.a. Insufficient/Improper input in the input fields; App prompts the user to correct these fields. 3.a. Post fails and app prompts user to try again later. Flow resets to step 2.

Use Case 10 – Access Calendar

Use Case Property	Description
Actor(s):	**TBA**
Pre-Condition:	**TBA**
Post-Condition:	**TBA**
Flow:	**TBA**

Use Case 11 – Calendar Reminders

Use Case Property	Description
Actor(s):	**TBA**
Pre-Condition:	**TBA**
Post-Condition:	**TBA**
Flow:	**TBA**

Use Case 12 – Calendar Group Invites

Use Case Property	Description
Actor(s):	• Users • Calendar API
Pre-Condition:	User is logged into the system and is on the calendar view of the app.
Post-Condition:	1. Member(s) are invited to the calendar group. 2. Member(s) are not invited to the calendar group.
Flow:	Normal Flow: 1. User indicates to invite other members to the calendar group. 2. User fills in members to invite. 3. User confirms form submission and submission is posted to the API. 4. App prompts user that specified members have been invited. Alternate Flow(s): 2.a. One or more members specified does not exist. App prompts user of member(s) that do not exist. 3.a. Post fails and app prompts user to try again later. Flow resets to step 2.

Use Case 13 – Posting to Facebook

Use Case Property	Description
Actor(s):	• User API • Facebook API
Pre-Condition:	User is logged into the system
Post-Condition:	1. Post of current page is shared 2. Post of current page is not shared
Flow:	Normal Flow: 1. User indicates to share the current page to Facebook. 2. App prompts user that the current page has been shared. Alternate Flow(s): 2.a. Post fails and app prompts user to try again later. Flow resets to step 1.

Use Case 14 – Posting to Twitter

Use Case Property	Description
Actor(s):	• User • Twitter API
Pre-Condition:	User is logged into the system.
Post-Condition:	1. Tweet of current page is shared 2. Tweet of current page is not shared
Flow:	Normal Flow: 3. User indicates to share the current page to Twitter. 4. App prompts user that the current page has been shared. Alternate Flow(s): 2.a. Post fails and app prompts user to try again later. Flow resets to step 1.

Use Case 15 – Access to Test and Quiz Material

Use Case Property	Description
Actor(s):	**TBA**
Pre-Condition:	**TBA**
Post-Condition:	**TBA**
Flow:	**TBA**

Use Case 16 – Flash Cards

Use Case Property	Description
Actor(s):	• User • Flashcard API
Pre-Condition:	User is logged into the system and is on the flash card view of the app.
Post-Condition:	1. Flash cards are edited. 2. Flash cards are not edited.
Flow:	Normal Flow: 1. User indicates to edit a group for flash cards. 2. User indicates to create/change a flash card. 3. User edits in front/back side of the card. 4. User confirms the details on the card are correct. 5. Card is saved. 6. App prompts user that the card has been saved. Alternate Flow(s): 3.a. Missing/Insufficient Content on the front or back side of the card. App prompts user for valid card content. 5.a. Post fails and app prompts user to try again later. Flow resets to step 3.

Lifecycle

The team plans to work using an agile software development lifecycle. During the requirements phase, features will be sub-divided into user stories and a rough estimate of team velocity will be used with respect to task delegation. Each task will be associated with a certain metric to determine/estimate task complexity. A number of tasks will be assigned to each team member in a sprint, a set period of development time. Each sprint, which the team has determined to be two weeks long, is expected to result in a working prototype of the features contained within the sprint. During each sprint, stand up meetings will occur every other day to track progress and address current issues. The team expects feedback and changes any time during the development process and these change requests will be discussed and handled accordingly amongst the team in terms of how the team feels is best for the project. A working prototype will be demoed to the client at the end of each sprint to exhibit progress and to solicit feedback from the client/other stakeholders. Sprint planning and task delegation occurs after these demos and feedback solicitation. This process repeats until the features agreed upon the team and the clients are fully functional and implemented.

User Environment

A user of this software can be reached from two different perspectives, as a student or as a teacher. Multiple users in completely different geographic locations should be able to interact with the Prof-desk system with an Android device similar to the web application. Depending if the user is a student or a teacher varies the amount of features that the application should perform, for example, a user logged in as a student should be able to select multiple courses from Prof-desk's course database and register to attend them. However, he or she should not have access to features given for a user logged in as a teacher like creating a course, providing study materials toward that course, and furthermore accessing a course's roster and gradebook. A typical log-in session needs to be secure, dependable, and overall user-friendly similar to many other Android applications where the user can be able to interact with the Prof-desk system as on the web.

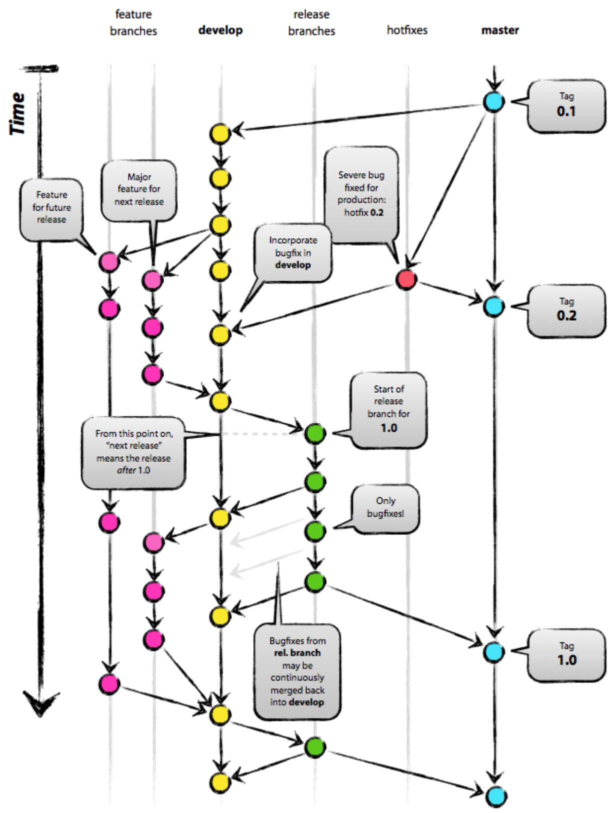
Configuration Management

This section will detail the procedures for source control management, project documentation, and project activity tracking.

Source Control Management

For source control management (SCM), our team will use the Git distributed version control system (VCS). Git is different from most VCS's in that we can create local and remote branches of the source code and make changes to those branches, minimizing the potential conflicts or other issues. Branches are quick and easy to create.

Below is a chart that summarizes our SCM activities using Git:


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

An explanation of the different branches in the above picture and how they will be used:

Branch Type	Description
Master	This branch will represent the system in a *production-ready* state. This means that the state of the source code on this branch can be delivered to the client to be used by all who use the system. By definition, merging into master means that we're in a new production release. When we merge into this branch we must also tag it with a release number (such as 0.3).
Develop	This branch is also known as the **baseline** branch. The state of this branch will reflect the system with the latest changes stated for the next release.
Feature	This branch represents a feature currently in development for a future release. It is created by branching off the **baseline** branch. When the feature is considered complete, it is merged back into **baseline.** There is no set naming convention except that it must not be **master**, **baseline**, **release-***, or **hotfix-***.

Release	This branch represents the system that is being prepared for production release. We branch from **baseline** and it is created when it reflects the state of the new release. We will use this branch to perform minor bug fixes or any other necessary changes. This branch will merge back into **baseline** and **master**.
Hotfix	This branch exists only when there is an immediate need to change the state of the system in production, such as a system-breaking bug. It branches from **master** and must be merged back into both **master** and **baseline**. If a **release** branch exists at this time, we merge into **release** instead of **baseline**.

We will use GitHub to remotely host our source code.

METRICS

The team plans to use time as the primary metric for measuring progress and tasks done. As mentioned above in the lifecycle section of this document, team velocity will be also used, with respect to time in our sprints, to measure progress and tasks done. More complicated features or user stories will be accounted in this metric to help the team estimate the amount of features/tasks completed that can be done in future sprints as the team gets more acquainted with the technology stack and codebase. Certain indirect measures may also be used to measure external attributes such as quality, complexity, efficiency, reliability, and maintainability as the project progresses.

USER INTERFACE

Please see the attached UI documents for the current beta interface.

NON-FUNCTIONAL REQUIREMENTS

USABILITY REQUIREMENTS

Ease of use in today's software is one of the top criteria for successful software projects. Here, ease of use conveys a measure of how easily a user can adapt to using software, how much training a user may need, and what type of training before they can use the software without looking at a tutorial or user guide.

- For example, a typical user should be able to navigate through the interface using your typical touchscreen gestures. The following gestures should be preferred while using the application if able to use Android's native OS features:
    - Ability to tap the screen to select the desired item
    - Ability to scroll up and down to view different pages in a document
    - Ability to swipe left or right to load the previous or the next document respectively
    - Ability to pinch the screen to zoom in and out (pinch fingers together to zoom in and pinch fingers apart to zoom out)
- The app should make as much use out of the native Android UI elements as possible to remain native and help the app flow smoothly. It should also be in line with Android's application standards within the marketplace.

- A typical user will have access to documentation on how the app works. Inclusion of a tutorial or user guide would be helpful.

Performance Requirements

At the following time, the client has not mentioned if there is a limit to the database size or the ease of access of creating an API to push and pull database information. Once this is certain, the performance requirements will be more elaborate.

Reliability Requirements

Obviously, the client would prefer that the system to remain functional as long as possible, barring any unforeseen issues with the system. Issues found within the app should be quickly fixed and produce minimal errors. With these criteria in mind, the system should try to meet the following reliability requirements.

- Assuming that the Prof-desk server is active and there is proper Internet connection available, the app should have no problems uploading changes to the server.
- There should be less than 50 unique bug reports for every 3 months of usage.
- Less than 1% of all bugs should be critical as defined by the user who submitted the bug report.
- If the app discovers that the system is down, it should notify the user that something wrong has happened to the server and not proceed any further.

Supportability Requirements

Since the developers will have to stop working on the project at the end of May, the app must be built in a way that allows it to be supported in the future.

- A bug report e-mail must be provided to submit bugs.
- A bug capturing system will take those e-mails and organize them.
- Requests that are sent to the server will be logged.
- If the app crashes, a crash information report will be sent to the server and logged.

Documentation, Installation, Legal and Licensing Requirements

To fulfill the requirements of our Senior Design Project, we are required to keep up-to-date information on the current design and state of our project. We are also expected to keep an up-to-date record of the work that we complete in engineering journals where we will thoroughly keep a record of all of our activities related to the project. We will also take detailed minutes of all meetings that take place.

As an Android application, our application has installation requirements of needing be able to be hosted on Google's Play store. This will allow for users to easily download the application to their devices without any hassle. In order to publish on Google's Play store, we must follow the guidelines found on the Android Development website's "Launch Checklist" (http://developer.android.com/distribute/googleplay/publish/preparing.html), however the sections that most concern us are the "Core App Quality Guidelines" (http://developer.android.com/distribute/googleplay/quality/core.html) and "Google Play Policies and Guidelines" (http://developer.android.com/distribute/googleplay/quality/core.html)

Design Constraints

Source	Constraint	Rationale
Schedule	Complete by end of May 2014	The application needs to be completed by the end of the senior year of current members.
Technology	Support Android version Ice Cream Sandwich and higher	The app needs to support any versions at or above Android 4.0.
Technology	Stability on most Android devices (ability to test on simulators and actual Android devices)	The app needs to be stable on typical Android phones and tablets
Economics	No budget for this project	This is a student project with no designed budget. The required development devices can be borrowed from the department or other students if need.
Environment	Internet Connection	The app needs to have internet connect to transfer data from and to the back-end database.
Environment	Encrypted data	Due to security concerns, data needs to be encrypted when being accessed and transferred.
Environment	Domain-Specific User Access	The application needs to behave different depending on the user's status.
Systems	Specifically for Android devices	The app is designed to be run on Android OS.

Risk Assessment

The team has assessed several risks with the project. Given the current codebase, the amount of time required to refactor existing functionality and allow for APIs to be used by the mobile application may be very large because such process has to be completed for each feature that will be on the mobile application that exists on the current website. In addition, a majority of the team does not have extensive experience with either the mobile or web technology stack (Android/PHP) that will be used in this project. The team anticipates a certain learning curve for both technology stacks, but it is likely that the mobile application will only be, at most, at feature parity with the current website.

Change Control

Request Handling

The group would prefer to have a request in form of an email that lists the name of the feature, the priority of the change, whether it is a request to modify an existing feature or create new one, and the reasoning for the request. Each request must be verified and approved by the client, the course instructor, and each development team member before consideration.

Changes Toward the System

The project team is open to any request made by the client; however, unless the changes are of “Critical” priority, later requests will be marked as “Low” or even “Very Low” priority and will be considered after all critical features are implemented.

Changes Toward The Artifacts

The team will handle the source control with GitHub. Each group member will clone the main Git repository, and create his or her branch for review before merging it with the master branch. Please see our “Source Control Management” section for more details on our GitHub management. Additional project artifacts like documentation will be written and edited by any team member, and all documentation will be modified throughout the project’s lifecycle in accordance with the client’s change requests