Project Plan
- Introduction
  - Overview
  - Deliverables
  - Assumptions
  - Constraints
  - Risks and Assets
- Management Structure
  - Project Affiliates
  - Project Lifecycle
  - Risk and Asset Management
  - Issue Management
- Planning and Control
  - Estimate
  - Tracking and Control
- Technical Process
  - Environment
    - Android development
    - Web service development
    - Development
    - Versioning
- Supporting Plans
  - Quality
    - Code standards
      - Java
    - User interface standards
      - Android Client
  - Testing

Project Plan

Introduction

Overview

With smoking being the leading cause of preventable deaths, there is a need for resources and programs to aid those who wish to quit. Smoking Cessation is an app meant to do just that by allowing for users to use Cognitive Behavioral Therapy to help them identify habits and triggers that encourage smoking and overcome or avoid them.

Smoking Cessation attempts to collect information in an intuitive manner weeks before a user begins their pathway to abstinence. With this data, helpful information and alerts are delivered to the user over time to remind them of their progress and where to move next. By relying upon mobile platforms, live data allows for the app to strive for "just in time intervention," or catching someone before they light the cigarette. Patterns of smoking activity can be observed over time to build behavior-specific suggestions and interventions for users.

SC's initial release will be on the Android platform for the general public. Future goals will include adding support for all major mobile devices as well as adding a complementary app that will allow psychologists to observe and support patients’ progress.

Deliverables

- Domain Model - model of the data in the relevant domain as it pertains to the user
- Project Process Document - a document outlining the development process
- Project Plan Document (this document) - this document outlining the project plan
- Project Website - website for the project containing information such as a description, time sheet, updates, etc.
- Project Schedule - Schedule for tasks and major deliverables
- Weekly Four Up - weekly update on what was done and what is planned to be done next, as well as risks and needs
- Effort Tracking Report - time/effort sheet of work done with metrics
- Interim and Final Presentations - presentation of the project to the sponsor
- Project Poster - poster describing the project
- Technical Report - report about the technical aspects of the project
- Team Self Assessment - team assesses their performance on the project
- Curriculum Reflection Report - reflection by the team on the project
- APK/Application (possibly in CD format) - product for the sponsor

Assumptions

- Users’ devices will connect to the Internet at least sporadically
- Users will have recent versions of the Android OS

Constraints

- App must support backwards compatibility for older phones
- HIPAA compliance
- The ability to add a web-service must be considered in development of mobile apps
- Limited battery life of mobile devices

Risks and Assets

Risks:

- Poor understanding of requirements - Minimal
- ~~Unable to complete essential parts of the system~~
- Falling behind on the project due to unforeseen circumstances - High
    - This has happened
    - Could be a result of not following process strictly enough
    - Not enough deadlines
- General lack of Android experience on team - Low
- ~~Security vulnerabilities that threaten HIPAA compliance~~
    - Irrelevant since we will not be addressing this
- Spending unexpected amounts of time on research
    - Alarm manager took longer than expect
    - Location
- Web-service may present unexpected complications
    - Syncing databases
    - API definition

Assets:

- Privately hosted GitHub repositories
- Slack Team chat room
- Development machines
    - Windows (1)
    - Mac (2)
    - Linux (1)
    - Lab machines (backup)
- Mobile devices
    - Each teammate owns an Android device
    - Sponsor owns Android tablet
    - One member owns an iOS device
- Team task management
    - ~~Trello (Non-technical goals / deliverables)~~
    - GitHub issues (Technical / code-based)
- Continuous integration server
    - Travis Pro
- Application server for web service
    - Undetermined

Management Structure

Project Affiliates

- 4 Software Engineers
  - Michael Dirmyer
  - Nick McCurdy
  - John Deeney
  - Michael Kenworthy
- Sponsor
  - Joseph S. Baschnagel, Ph.D.
- Coach
  - Al Arujunan

Project Lifecycle

The project’s lifecycle will follow the spiral process model. This model was chosen for its flexibility with large projects. The model allows for the project to be split into phases which each contain their own set of defined requirements, risk analysis, planning, and documentation. Splitting the requirements into phases also provides the benefit of mitigating risk of the requirement changes that are inevitable.

The implementation of the model for this project will involve seven phases*. During each phase, the requirements and risks relevant will be identified and solidified and separated out. Afterwards, the team will plan out how the code should be designed/structured. During and after development, tests will be written to validate functionality being implemented. Once written, the code will be deployed (not necessarily to the public). Once completed, a post-mortem will be held to identify the good and bad practices observed within the phase. Finally, documentation and code will be reviewed and presented to the project sponsor and coach for approval.

*The specific details of each phase are detailed in the separate document Development Phases

Risk and Asset Management

Risks will be assessed at the start of each phase and will be recorded within the weekly four-ups. Major risks will be recorded within the project plan and reviewed with the sponsor on a weekly or bi-weekly basis, depending on needs.

Issue Management

Code-related issues and bugs will be tracked, triaged, and discussed within their respective GitHub repositories. Existing issues will be reviewed each week. At review time, the risk of the particular issue should be assessed and a plan should be discussed for resolving the issue. Once a plan has been constructed, it should be assigned to a member alongside new feature development deliverables. If an issue is high-priority and is blocking the development of a new feature or resolution of an existing non-technical bug that could result in a change of project schedule, it should be brought to the attention of the sponsor within a week of creation.

High-level or non-technical issues and bugs will be tracked within the team’s Trello board. These issues may relate to sponsor feedback/requests, new feature development, or new constraints that need to be addressed.

Planning and Control

Estimate

Describe any and all estimates to date relating to the project and what will be done to refine the estimate during the project.

There will be estimates on how long each phase will take. Each phase will have an estimation from the beginning. They will be re-estimated during the risk analysis of each phase.

Phases:

1. Initial estimate (Fall Sem 2015)
    a. 3.5 weeks
    b. 3 weeks
    c. 5 weeks (candidate for splitting)
    d. 4 weeks
    e. 6 weeks (candidate for splitting)
    f. 2 weeks
    g. TBD
2. Second estimate (Spring Sem 2016
    a. 16 weeks
    b. 6 weeks
    c. 4 weeks
    d. Out of scope

Tracking and Control

High level deliverables, tasks, and activities will all be tracked and assigned within the team’s Github issues. Expected due dates, involved members, and details about the tasks will be added to each issue.

A requirements document will store all created and agreed to requirements. Requirement details and progress will also be stored here. Upon implementation and verification of a requirement, it will be marked and slated to be reviewed with the sponsor. After validation, the requirement will be marked completed and archived for future reference. At the end of each phase, completed requirements will be reviewed as part of the teams post-mortem.

Code changes will be tracked via GitHub pull-requests. All code-changes will be made within its own pull-request and will not be merged into the codes master branch until reviewed by at least one other developer.

High-level team progress will be tracked within weekly four-up charts. These will contain overviews of each members progress relevant to the senior project coach.

Questions and concerns related to the project will be discussed within Slack. If necessary, “Slack reactions” will be used to obtain approval for decisions related to discussions.

Technical Process

Environment

Android Client

Application development and build configuration for the Android application will be completed within Android Studio IDE. Builds and related tasks for this operation system will be automated using Gradle. Gradle will be available in local and testing environments.

Automated builds and testing for the Android app will be completed for the master branch as well as all branches associated with pull-requests within GitHub. This automation will be configured and completed using Travis Pro.

Once a release has been agreed upon, the application will be reviewed with the sponsor. Sponsor permitting, the app will be deployed to the Google Play store.

Web service

The stack that will be used for the web service is undecided. Due to that, we also have not decided on the tools that will be used for code-development and build management.

The system architecture, including the server requirements or hosting requirements, operating system, and other operating software, has also not been decided on.

All developed code will be saved within a separate repository from the Android project with similar build and test configurations.

Development

After checking out the repository to be changed, members will branch from the master branch. Their new branch will be a dashed branch describing the change to be made. Once branched, a pull-request will be created.

Shortly after beginning development, members should begin code-reviewing peers pull-requests. Once a change has been completed, the developer will request for a final review. A thumbs up emoji (“:+1:”) within the pull-request placed after the final commit and a passing build will symbolize a completed set of changes and that it can be merged into master.

NOTE: The master branch should never receive a direct commit. The only exceptions being changes to readmes or documentation changes which do not need review.

Versioning

As needed, labels will be added to commits to represent various versions of the applications

Overriding the process

Any time a derivation must be made from this plan, it must be stated within the Slack channel or discussed at a project meeting. If the derivation could impact the sponsor or have any financial implication, it must be brought to the attention of the sponsor.

At the first reasonable time, the occurrence must be documented within table 1 at the end of this document.

Supporting Plans

Quality

Throughout the development process, architectural diagrams, wire-frames, mock-ups, documentation, and other materials will be used to keep the sponsor informed about the details of the application. Feedback from the sponsor will influence decisions made regarding each piece to ensure the proper product is delivered.

Research will be done to understand how to best engage users within the scope of the product. This research will strive to improve information gathering techniques and interface design. User surveys will be developed to obtain feedback from the sponsor and others interested in the project. One considered group will be the staff at the RIT counseling center.

Code standards

Java

For this project, we have decided to use Google’s Java standards. We will use a linter to enforce this.

User interface standards

Android App

Material Design Light (with custom branding colors)

Testing

All

Appropriate test code should be written as new features, fixes, and feature changes are developed. Pull requests on significant (non-trivial) code changes should have unit and/or acceptance tests for most added and changed code. It’s your responsibility to have tests that are accurate enough to represent when someone else breaks your code.

Android App

The full build and test suite will be run on developers local machines. We have decided to avoid using the continuous integration technology for testing as the extra steps necessary to get this testing running with Active Android are not simple. Besides automated tests, our Android app may eventually be tested with smoking users at the student health center on campus.

Web Service

Web services should have periodic stress testing, with details decided later in the project. We may also plan on setting up continuous delivery (automatic deployment) for the web service later in the project.