Project Plan

for

SISCalendar

Prepared by Zach Masiello
Ethan Mick
Michael Caputo
Shawn Thompson
Organization: SIS.io

Revision History

Name	Date	Reason For Changes	Version
Initial	10/1/13	First version	1.0
Revision 1	10/3/13	Updated calendar and added roles	1.1
Revision 2	10/8/13	Updated ITS deliverables	1.2
Revision 3	10/12/13	Updated schedule. Divided technical tools into server and client sections.	1.3
Revision 4	10/24/13	Added several new process metrics.	1.4
Revision 5	1/30/14	Updated for Scrum scheduling	1.5
Revision 6	4/16/14	Updated schedule	1.6
Revision 7	4/20/14	Updated Methodologies and Project tracking	1.7

1. Overview

Information & Technology Services updated the Student Information Service this past year, which is the system that allows students to access their class schedule and choose their classes. The new system offers many improvement over the old system, but is still missing some crucial functionality. Student Government sent out a survey asking students what functionality they would like to see added. The biggest requests were; the ability to export the class schedule to Google Calendar/iCal, and the ability to view class locations on the RIT map.

Team SIS.io will add this desired functionality to the SIS system through a satellite service. This service will be implemented as a responsive web application, allowing devices of all screen size to easily access the functionality. The service will use the campus' central authentication system to ensure users only access their own information and will utilize ITS' application programming interface in order to provide the data.

This service can be used by students on all of the RIT campuses. In this case, a student is any person taking classes at RIT. If the student is taking classes on RIT's Rochester campus the service will allow the student to view their classes through the RIT campus map; however this will not be enabled if the student does not have classes on that campus.

The project must be completely finished by May 22nd, the last day of finals. However, SIS.io will break the development into three main versions, the first of which will deliver all major functionality before the end of Fall semester. Version 2 and 3 will be worked on and released in the Spring semester, building on version 1. This will allow for early feedback to be incorporated in later releases.

SIS.io is responsible for building the entire web application. The service will be responsible for authenticating and getting information, but SIS.io is not required to build the services that provide that data. SIS.io will write all documentation required for a smooth transition and will develop the tools required for deployment. Actual deployment will be handled by the ITS team.

2. Goals and scope

2.1. Goals

- Students will be able to export their schedule into their google calendar or to an iCal file.
- Students will be able to see their class' building location on the RIT map

2.2. Scope

2.2.1 In Scope

- Getting the data from the ITS API and converting it into an iCal or importing it into RIT-Gmail
    - UnderGrad and Graduate classes
    - For all of RIT's Campus'
- Using RIT's Shibboleth system for user authentication
- Displaying class locations on the RIT Map
    - Only for the Rochester Campus

2.2.2 Out of Scope

- Getting the data from SIS. ITS will be providing an API to get us the data.
- Interfacing with RIT Maps, ITS has provided an API for this.
- New buildings being added to RIT Maps

3. Deliverables

The deliverables for the project will be split between those for the Software Engineering department and those for ITS.

3.1. SE Required Deliverables

- Project website holding all non-proprietary work products and project artifacts maintained in the project account on the ***se.rit.edu*** web server. If for some reason, you are hosting project artifacts on another web server, a static mirror image must also be maintained on the department server.
- Project plan, schedule and process methodology definition prepared by the end of week 4/3 (fall/summer start) of the first term.
- Tracking report for time/effort worked by each team member and the team aggregate updated on the project website weekly. Tracking report for at least two product/process metrics appropriate to the project and development methodology updated on the project website at least every two weeks.
- Make interim status and final project presentations. Attend presentations given by other teams and provide constructive feedback on the presentations.
- Project poster and presentation at "SE Senior Project Day"
- Project technical report
- Interim and final team self-assessment
- Post-mortem curriculum reflection report
- A CD(s) at the conclusion of the project containing all project artifacts.
- Each team member completes a Software Engineering Program Senior survey.

3.2. ITS Required Deliverables:

- Requirements
- Functional design document
- Technical Approach and overall Design after Proof of Concept, but before development begins
- Wireframe Review
- Technical Design Review
- Testing Approach
    - Includes plans, scripts and approach for creating test accounts
- Deployment to TEST using EWA script process
    - Includes installation instruction specific to our development.
- Technical Review of application
- Deployable system with minimal functionality (iCal export and RIT maps integration)
- Support handoff documentation
- User Manual

4. Roles

- Project Manager - Ethan Mick

The project manager is in charge of ensuring smooth communication with the client team and productivity within the team. If any issue arise, the PM needs to deal with them efficiently and effectively. The PM is also in charge of managing risks, tasks, and timing.

- Requirements Manager- Shawn Thompson

The requirements manager is the master of all things relating to requirements. As the project progresses, changes may need to be made, or requirements will need to be managed. The requirements manager handles all of these requests and ensures the project can stay in scope. The requirements manager will also scribe the meetings.

- Architect - Mike Caputo

The architect is the master of design and technical overview. He understands the technology stack, knows the positives and negatives about the stack and architecture, and fully understands how the system works and interacts with other systems.

- Test Lead - Zach Masiello

The test lead manages all aspects of testing. He understands how the tests are put together, goes through test coverage, ensures complicated parts of the system are well tested, and can put together reports of testing. He works closely with the architect to ensure the system is testable, and also works closely with the requirements manager to ensure the acceptance tests are met.

All roles are empowered to update the website as necessary.

5. Risk Management

In this section, we have identified some risks that may present themselves throughout the course of this project. We have also identified warning signs that a risk is occurring as well as mitigation strategies for protecting our project from these risks. The “Assessment” section of each risk is used as a ranking for the priority of a risk, calculated by probability of the risk occurring multiplied by the impact if the risk were to occur.

5.1. Project Risks

Please see the Risks Document.

6. Scheduling and estimates

6.1. Overall project schedule

Our project schedule extends 2 RIT semesters. Meetings began in September, with R1 officially beginning at the end of September. The schedule then accounts for three different releases, culminating with the final delivery of our product in mid May.

6.2. Work breakdown

- Student authentication
    - Integration with Shibboleth
- iCal generation
    - Integration with ITS API
    - Downloading iCal
- RIT Map integration
    - Collection of data
- Document deliverables
    - Project plan
    - Use case document
    - Technical approach
    - SRS document
    - Design document

    - Wireframes
    - Test plan
    - Application Files
    - Hand-off document
    - SE specific documents
        - Technical Report
        - Project Poster

6.3. Definitive schedule

6.3.1. Release 1

09-24-13
- Project Plan (Draft)

10-01-13
- Use case document (Draft)
- Set up Dev environment with GitHub

10-08-13
- Query about GitHub ITS account
- Incorporate feedback for Use Cases
- Project Plan final Review with ITS
- SRS (Draft)
- Technical approach review
- Schedule following week's meeting

10-17-13
- Incorporate feedback for SRS
- Use Case Document (Final)
- Project Plan (Final)
- Design Document & Wireframes (Draft)
- Feedback on designs

10-22-13
- SRS (Final)
- Incorporate feedback for Design Document & Wireframes
- Begin Development Work (authentication, non-API work)

10-29-13
- Design Document & Wireframes (Final)
- Continue Development (authentication, non-API work)
- Begin work on Test Plan

11-5-13
- Test Plan (Draft)
- Development Meeting w/Lisa
- API access for eServices information and RIT Maps Application

11-12-13

- Review if we need access for test server.
- Customer demo

11-15-13

- Request Test Environment

11-19-13

- Test Plan (Final)

11-26-13

- No meeting, Thanksgiving week

12-3-13

- Get test server around here
- R1 Deployed to test
- Request Production Server

12-10-13

- 4:00pm - Interim presentation
- Final meeting of the Semester

12-13-13

- R1 testing completed
- R1 code frozen and ready for production

6.3.2. Release 2

1-28-14

- First Meeting of the 2nd Semester
- Plan out scrum

2-4-14

- Meeting with Lisa
- Explore possible automation
- Ask about Database
- Setup Dev/Test Environment

2-10-14 (Sprint 1)

- Dynamically Add End Dates to Classes
- Include Professor Contact information

2-17-14 (Sprint 2)

- Ethan is not going to be here
- Cache Holiday Calendar Data
- Pick the Term

2-24-14 (Sprint 3)

- Help Desk Document

- Support Document
- Cache Class Location
- Instructions for iCal Handling
- Hide Withdrawn Classes
- Add Schedule directly to Google Calendar

3-3-14 (Sprint 4)

- Add Schedule directly to Google Calendar
- Instructions for iCal Handling →
- As an unauthorized, but authenticated, user, when I log in I should see an information page.

3-10-14 (Sprint 5)

- As a user, I should see holiday schedule information on my calendar.
- Add Schedule directly to Google Calendar
- As an Administrator, I would like to see Google Analytics for the application.
- As a user, I want to see my final exams.

3-17-14 (Sprint 6)

- As a user, I should see holiday schedule information on my calendar.
- Add Schedule directly to Google Calendar
- As an Administrator, I would like to see Google Analytics for the application.
- As a user, I want to see my final exams.
- Bug Fixes

3-23-14 (Sprint 7)

- R2 Complete
- As a user, I should see holiday schedule information on my calendar.
- Add Schedule directly to Google Calendar
- As an Administrator, I would like to see Google Analytics for the application.
- As a user, I want to see my final exams.
- Bug Fixes

3-31-14

- Spring Break

6.3.3. Release 3

4-7-14 (Sprint 8)

- Bug Fixes
- Code Walkthroughs (Prior to, technical documentation given 1 week)
- Update code from code review.
- Senior Project Poster

4-14-14 (Sprint 9)

- Bug Fixes
- Code Walkthroughs
- Testing and User Testing
- Senior Project Technical Report

4-21-14 (Sprint 10)

- Bug Fixes
- Code Walkthroughs
- Testing and User Testing
- Senior Project Technical Report
- 4/24 - Meeting to review data and application.
    - 1-2, building 1.

4-28-14 (Sprint 11)

- Code Freeze
- Only Critical Bug fixes and refactoring
- All documents finalized
- Senior Project Technical Report

5-5-14 (Sprint 12)

- Go /No Go?
- Push to Production
- Only Critical Bug fixes
- Senior Project Artifacts

5-12-14

- Senior Project Artifacts
- Senior Project Presentation

5-19-14

- Graduation

6.4. Resource allocation

Each member of our team is expected to put in 8-10 hours per week. This will be tracked

following the description found in the Metrics section of this document. Failure to do so may lead to risks, and as such any instance of a team member failing to put in their time will be corrected as outlined in the Risk and Risk Management section of this document.

6.5. Estimation techniques

Each member of the team will be expected to estimate their own portions of the work and track those estimates within JIRA. Updates to all estimates will also be the responsibility of individual developers.

6.6. Project tracking

SIS.io will track progress through issue cards. We will use Trello as our issue tracking tool. This will allow us to track times, estimates, bugs, and user stories.

Trello will be able to track all issuesby using a “Known Bugs” list. Bugs can be taken from the Known Bugs list and added to a sprint, at which time the person assigned to complete that bug may move the bug card into the “Doing” list when they begin working, and the “Sprint X Done” list when they fix the bug. Each developer will be responsible for adding a comment to the Trello card upon completion with the total number of hours spent working on that bug.

The same process will be used for functionality. Each piece of functionality to be implemented will be added to a new Trello card in the “Backlog” list. Each functionality card will also be connected to a parent User Story card (also in the “Backlog”). As cards are pulled into a sprint, the card will be moved into the “Sprint X” list. When work begins on that functionality, the developer implementing the actual functionality will move the card to “Doing” and then to the proper “Done” list once the functionality is completed. Each developer will be responsible for adding the amount of time they spent on a task to each task card.

6.7. Schedule Changes

Any schedule changes required due to missed deliverables will require scope changes that must be discussed and approved by the RIT ITS team.

7. Measurements & Metrics

Comment Density: The number of CLOC (Comment lines of code) / LOC (Lines of code). This will be used to indicate how well the source code is documented.

Cyclomatic Complexity: The number of linearly independent paths through a functions source code. This will be used to indicate the complexity of each of the functions within the project.

Defect Density: The number of defects per line of code. This metric will be used to determine how error-prone our project is.

Test line coverage: The percentage of total lines of production code covered by automated tests at any dimension (unit, integration, graphical, etc).

Hours of effort: The total number of hours, doing any work for senior project, per developer.

Google Analytics: Generate detailed statistics about the web application's traffic and traffic sources.

Jira tasks closed per individual: The number of tasks per week that a team member successfully closes.

Average time Jira task is open: The average amount of time an individual team member leaves open Jira tasks.

Number of late tasks: The number of tasks per week that a team member delivers "late" - where late is defined as Sunday night or later before our Tuesday meeting with ITS.

Trello tasks closed per individual: The number of tasks per sprint that a team member successfully closes.

8. Technical Process

8.1. Development Methodology

Evolutionary Delivery

Evolutionary Delivery is a Methodology that blends Evolutionary prototyping and Staged Delivery. In evolutionary prototyping the most prominent parts of the program are built first and then showed to the customer. The customer provides feedback, and then the next prototype is worked on. This loop continues until the prototype is essentially "good enough," and is released as the final software. Staged Delivery is similar to evolutionary prototyping, but doesn't rely on customer feedback nearly as much. Rather, the approach incrementally builds the software and delivers it to the customer until finished.

Evolutionary Delivery blends the two together, allowing for a lot of flexibility in how much feedback is incorporated into the next version. This is fantastic for SIS.io, because the first version of the product has rather straightforward requirements, the amount of feedback and changes should be minimal. However, version 2 and 3 are much more loosely defined, and having feedback from the customer will be critical in delivery a high quality product which satisfies their requirements. By using the Evolutionary Delivery methodology for building this

software, we will be able to incorporate feedback as necessary.

We will be using JIRA to track our tasks, using Toggl to track time worked on tasks. There are numerous documents that will need to be maintained, but the scope of version 1 will keep these documents small. These documents will include requirements, design, and test documents. These documents will be updated throughout the project and delivered at the end. Please reference those documents for additional details.

Scrum

Scrum is an iterative and incremental agile framework that utilizes a flexible development strategy to ensure functionality that the customers care about is delivered on time and to the customers satisfaction. The lifecycle of Scrum is broken into chunks called “Sprints”. SIS.io will utilize one-week sprints to ensure that work is not put off until the last minute and thus wasting one or more weeks of development time.

Our implementation of Scrum will begin each week on Monday. The SIS.io team will meet each Monday for a weekly reflection and sprint planning meeting. At this meeting, the team will review what went well during the last sprint, what problems were encountered in the last sprint, and what could be done better in the upcoming sprint. After this reflection, the team will discuss any necessary changes to the methodology that are required based on the reflection feedback. Should any changes be required, they will be documented in the following section, “8.2. Methodology Changes”.

Once any new changes to the methodology are settled, the team will begin with the sprint planning portion of the meeting. This part of the weekly meeting will involve looking at the Backlog list in Trello and determining which high priority tasks to pull into the sprint this week. Tasks will be pulled into a sprint based on their priority as determined from the following criteria:

1. The priority of the task to the customer
2. The amount of time the task is expected to take
3. The amount of work that has already been put into the task

After discussing the above criteria, if their are any ambiguities of what the task involves then the task will be postponed until clarification from the customer can be obtained. The number of tasks pulled into a sprint will vary week to week based on the velocity (the number of hours the team completed in the last sprint). The amount of hours in a sprint should not exceed the amount of hours completed in the last sprint.

Each Tuesday, the SIS.io team will have a customer meeting. During this meeting, we will provide a demo of any new functionality to SISCalendar that was completed during the previous sprint. SIS.io will also update the customer on everything that was completed in the previous sprint, what things were not completed in the previous sprint, any ambiguities in the backlog, and

what is planned for the upcoming sprint. Customer feedback throughout this meeting will be noted in meeting notes posted to the SIS.io website and will be added to the SIS.io team's backlog as feedback to discuss at the next sprint planning meeting.

Another important aspect of the scrum methodology is the daily standup. The daily standup is a meeting between all developers in SIS.io each day to discuss what each person accomplished that day, what that person expects to accomplish tomorrow, and any blockers that the person encountered during the course of their work. Due to the schedules of the SIS.io team, daily standups will be held every day (with the exception of Monday and Tuesday - the sprint planning and customer meetings will be counted as daily stand ups for each of those days respectively) at 9pm. The standups will be held using GoogleHangout.

8.2. Methodology Changes

Change 1: After winter break, the SIS.io team will transition from an Evolutionary Delivery methodology to Scrum. This will be accomplished because all of the required functionality of SISCalendar has been built and is ready to be delivered to the customer. As such, for semester 2 the team will use sprints to ensure that at the end of each week a functional product that could be delivered as is can be shown to the customer. Sprints will also make it easier to ensure that the smallest amount of time possible is spent on functionality before getting feedback from the customer so that appropriate changes can be made.

Change 2: After the second sprint in Semester 2, the SIS.io team has decided to change the daily standup from using GoogleHangout's to using a GroupMe group text message. This decision was made because not everyone had access to the proper computer equipment to have effective GoogleHangout's and too many members of the team were unable to be at a location that allowed for a GoogleHangout at 9pm at night.

8.3. Tools

There are a number of tools and technologies that this project will utilize for development.

- Server Side
    - Java 1.7.0_40 environment
    - Apache Struts 2.3.15.2 framework
    - Apache ANT
    - Eclipse Kepler 4.3 IDE
    - Git repository (on Github)
- Client Side
    - Jquery 2.0.3
    - normalize.css

We have decided to use the Struts 2 framework for our development for a number of reasons. Struts 2 is the most up-to-date (stable) version of the Struts framework currently released by

Apache (updated as of 09-21-13). This means that problems stemming from the framework itself should be minimal. Struts is also one of the approved technologies given to the student team as it is currently already supported by ITS in the Team Builder project. The Team Builder project has already solved the problem of Shibbeloth authentication in the Struts 2 framework, which will allow the SIS.io team to reuse code shortening the expected schedule and avoiding unforseen challenges caused by Shibbeloth authentication with new technologies.

Struts 2 also provides a Model-View-Controller paradigm to applications developed in it. This will allow the SIS.io team to create a clean, reusable, and easily extendable application within the confines of the schedule.

Apache Ant will be used in the creation of .war file builds for development, testing, and production. Ant is one of the ITS approved libraries for creating java builds for deployment, as well as being the library used by the Team Builder application. SIS.io will be able to utilize Team Builders Ant scripts as examples when constructing our own, making the process as a whole easier.

Development for SISCalendar will take place on each students own personal computer. Eclipse Kepler 4.3 shall be used as the integrated development environment. Eclipse was chosen because it has plugins that will allow easy integration for both the Struts framework and also Apache Ant to aid in the development and deployment processes.

SIS.io will use a git repository to store our code. A master branch will be stored on the account provided by RIT ITS, and each SIS.io developer will maintain their own local copy and feature branches.

For rendering and interacting with the client side, SISCalendar will utilize JQuery and normalize.css. JQuery will allow the manipulation of data and UI elements easily and efficiently on the client side. Normalize.css will be used to ensure SISCalendar has cross-browser consistency when the UI is rendered.