Prepared for

Statement of Work for

Grand Tour Messenger

Date	Version	Author	Modification
Jul 23, 2012	1	Jack Flannery	Initial Document

1. Purpose

Grand Tour Messenger (GTM) aims to provide Amtrak's mission critical employees with a modern communications platform. The goal is to improve the current state of communication between Amtrak's employees. Employees need to communicate information fast, reliably and efficiently and they need a tool that will make this easy, and not get in their way. Information must be passed along quickly so they can get on to their next task. GTM will make it's users more productive and make their jobs easier.

2. Requirements

Grand Tour Messenger should allow users to send instant text messages to other users. GTM must also support sending files to other users. GTM must support continuous operation at six user workstations. Those workstations are named the following:

Chicago Union Station North (CUSN)
Chicago Union Station South (CUSS)
Amtrak Michigan Line / New Orleans (AML)
Yard Control (YDCTL)
Yard Master (YDMSTR)
Glasshouse (GLHS)

GTM users should be able to choose a workstation when they sign in (regardless of their physical machine). Only one user may be signed into a workstation at a time. When a user is sending messages, they will choose the recipients of their messages by selecting one or more workstations out of the six available. The user(s) signed in at the selected workstations should be the recipients of the messages. More often, users will be concerned about what workstation the messages are sent to, and not so much which user.

When a user signs into GTM, they should see any recent messages sent to their workstation while no user was signed into it. They should not see any recent messages sent to their workstation while other users were signed in it. However, when a file is sent to a workstation, all subsequent users

signed in at that workstation should have access to the file.

3. Implementation Details

Grand Tour Messenger is implemented as a standards compliant web application on a private Amtrak network. The web server will be provided by Grand Tour Software and will be located at The Chicago Control Center. User consoles will be provided by Amtrak and will be located at the six workstation locations listed in section 2. GTM requires only a web browser for a user console to be a GTM workstation. No additional software is required on the consoles.

GTM will allow users to register a user name and password with the system. Each password is stored in the database as an encrypted salted hash. Initially, there will be one administrator user with added privileges such as the ability to create and view transcripts and manage users. The administrator can later elevate other users to be administrators. To sign into GTM, users must provide their user name, password, and choose a workstation out of the six available workstations. Administrator users are not required to choose a workstation. Only one user is permitted to be signed in at a workstation at a time. At registration, users will also be asked which workstation they work at on a daily basis in order to save them the time of selecting the same workstation every day.

When a user signs in, they will be directed to the messaging page. The messaging page is where users will receive, compose and send messages. Files can also be sent and received from the messaging page. The messaging page has six toggle buttons, one for each workstation. The button will show the name of the workstation as well as the user who is currently signed into that workstation. When no user is signed in at a workstation, it will say 'vacant'. The button text is updated in real time, as users are signing in and out, so that the button always shows the name of the user working at the workstation. When a user clicks one of the buttons it will toggle on, changing in appearance, indicating that the user is now messaging that workstation. All subsequent messages sent from the user's console will be routed to the selected workstation and received by the user signed in at that workstation, or the next user to sign in at the workstation if no user

currently is. Another click of the button will toggle it off, and subsequent messages from the user will no longer be routed to that workstation. A user may select one or all workstations at once to send a message, or any combination. There is an additional 'Message all' button that when clicked will toggle all workstations on, and a second click will toggle all workstations off. Pressing the reload button on the browser will reload the messages on the page, bringing back the last 24 hours of messages.

When a message is received it will indicate that the message is unread. When the receiving user clicks the message it will change color indicating that it has been read. A message will also be instantly pushed to the sending user, informing them that their message has been read.

An administrator is able to view and create transcripts. A transcript is created by entering a date range and selecting a user and/or workstation. The transcript will show all of the messages in the selected date range that were sent to and received by the selected user or workstation. The transcripts will be saved in the system and can be viewed, modified, or deleted later. An administrator may create and save a very large number of transcripts at once. When a transcript is viewed on the screen it will be formatted for the screen. When a transcript is printed it will have alternate formatting, optimized for the printed page.

4. Technologies

Grand Tour Messenger (GTM)is implemented as a web application. Communication between the clients running the Chrome web browser and the web server is through standard HTTP and AJAX. Web sockets are used to push new messages from the server out to the clients. The server runs on the GNU/Linux operating system, specifically Ubuntu Server Edition version 12.04. The Nginx web server handles all HTTP requests and serves static content and files. The application is written in standards based HTML, CSS and JavaScript on the client side. The Ruby programming language along with the Ruby on Rails framework is used on the server side. Data is persisted to a Postgresql relational database on the server. GTM uses Git for source control.

5. About Grand Tour Software

Grand Tour software is currently a one man software development shop. It's core principal is the Agile software development methodology. The agile movement was started back in 2001 and means many things to many people. Traditional thinking in software development is that as software grows and becomes more complex, it becomes increasingly more difficult and more expensive to change. To combat this, many software development teams spend more time planning and analyzing requirements than they do actual programming. Once the initial programming is done a team of testers is required to manually test the application. And any changes thereafter require the full application be retested for regression errors.

Grand Tour software is developed using strict test driven development (TDD) and Behavior Driven Development (BDD). In test driven development, sometimes referred to as test first development, before any application code is written, a small test program (or unit test) is written to test the code that is not yet written. A unit test is intended to test a very small granular unit of code within the application. A unit test typically works by executing a unit of code and tests for the correct outcome. When a test is written, and is run, it will initially fail because the application code has not yet been written. The next step is to write the application code that makes the test pass. Once you have a passing test you can review the application code that you just wrote and possibly make improvements to the organization of the code to ease future maintenance. You can then rerun the test to make sure that it still passes and be sure that the organization improvements didn't break anything. Then repeat the process. Write a test, make it pass, and refactor the code. As your code base grows, so does your test suite. TDD has many benefits. First, it encourages you to write code in small testable units. Second, they serve as a safety net. The test suite should be rerun often during development to ensure against regression. Thirdly, the tests serve as documentation to future programmers of the system. The unit tests specify exactly what each unit of code should do.

Behavior Driven Development takes this idea further. In BDD, we do test first development with what are known as integration tests. Integration tests test the entire application stack, from the perspective of a user.

Grand Tour makes use of the popular open source tool known as Cucumber. With Cucumber, tests are written in plain English that any stakeholder of the project can understand. Cucumber tests are executable and simulate a user using the application. Each Cucumber step is mapped to an executable line of code that is capable of simulating a user by automatically driving a web browser and then testing for the correct outcome. Grand Tour Messenger is developed using the BDD cycle. First a large Cucumber integration test is written. This test will describe an entire feature that could be well understood by any user of the system. Then in the process of getting that integration test to pass, several unit test are typically written which specify all of the individual units of code required to implement the feature. And the cycle repeats with next feature.

BDD and TDD lead to agile software that is able to respond well to changes. It is only natural for software requirements to change over time, so it is very important that the software be malleable and easy to change. Having a code base with solid test coverage can make that possible.

Cucumber tests are an integral part of Grand Tour Software. All features delivered will be accompanied by Cucumber tests that can be read and understood by all project stakeholders. Aside from being executable, Cucumber tests serve as documentation that describes the behavior of the application in detail. Future changes and enhancements to the system will be delivered on a feature by feature basis, and not in large periodic releases, and will always be accompanied by Cucumber tests.