Order of PI: Webscale

Requirements Specification

November 22, 2016

0xf Solutions ltd.

Jakob Roberts
Jonah Boretsky
Edgardo Jose Cuello
Chris Kelly
Justin Richard
Jim Galloway
Jason Syrotuck
Konrad Schultz

Table of Contents

1 Introduction

As per the previously submitted Progress Report, the Engineering Student Society is visited annually by monks who belong to the Order of PI. These monks have aided the Engineering Student Society in collecting donations. The monks require certain items and services to continue to visit the engineering students. The current system used by the monks is too slow to process the number of incoming payment requests during the month of March. Furthermore, there is no guarantee of integrity between payments and donations. For this reason 0xf Solutions ltd. has proposed a complete rewrite of the system.

1.1 Purpose

The purpose of the system is to collect, record and track donations made to the Monks of Pi. It will also provide functionality for users to make donations. The system shall be a web application and users will have access to all the required information to learn about the Order of Pi, and lay charges to those deserving. Users will have the ability to

- donate to schedule PI'ings (hereafter referred to as "trials"),
- view available times for trials
- and make unrelated donations.

The system will give the Monks of Pi a way of easily seeing when bookings have been made, assigning volunteers to bookings, and ensuring that no double bookings are made. This will improve upon the current system, where most of this is done manually by the Order of PI.

1.2 Features

The booking system will provide end users with a way of creating, rescheduling and canceling trials. The application will also accept donations, and let users designate the time of the trial by displaying the current availability. The system will automatically designate volunteers for specific trials, based on their availability, which will be designated through a section of the website only available to volunteers and administrators. Finally, it will be used to obtain and aggregate voluntary user feedback, in furtherance of improving the system.

1.3 Outline

The purpose of this document is to outline the features of the project, and give information regarding the planning and implementation of the system. The document will serve as a high-level technical description of the proposed system. The following section describes the system in more detail, and gives more information regarding the current system and its limitations. Following this is an operational description, which includes a high level system model and some general ways in which users will interact with the system. We then describe the functional and nonfunctional requirements involved in the system. We conclude by giving insight into the feasibility of the proposed solution, and give a graphical model of the main components involved in the system.

2 Current Problem

The main problem this solution solves is the shortcomings of the current donation system. It will enable users and administrators to save time and have a better experience every March. This system will be able to scale to support significantly more users and be maintainable. This solution plans for the future and attempts to solve the aforementioned issues of the Order of PI.

2.1 Factors Affecting Requirements

There are many factors than can affect requirements. The general factors that are taken into consideration for these requirements are the stakeholders' demands, the expected number of users, the system limitations, and system configuration options. All of these factors impact both functional and non-functional requirements (e.g., the number of users directly affects the need for scalability).

2.2 Stakeholders

When building a system of this magnitude many parties are involved and interested in the outcome. The client for this project is the UVic Engineering Student Society. The monks of PI are a key stakeholder, as the system is specifically designed to support their activities. The intended users of the system consist of the monks of PI, Engineering Student Society,and students of UVic.

2.3 Constraints

Important constraints for this system include scalability and maintainability. It is important to the Engineering Student Society to future proof the system through scalability. The provided solution should also be maintainable to allow future alterations with ease. One more important constraint for this system is the interface between the scheduling and donation subsystems. It is also important to take into consideration hardware and platform constraints. The frontend applications will run on a web browser (Edge, IE, Chrome, Firefox) or mobile operating system (iOS, Android). They will be limited to supporting the latest version that supports 80% of users. Security will not be compromised for compatibility with older versions. The applications that the user interacts with are so far above the hardware that hardware constraints should not be an issue. The backend will be designed to run on a linux system. One of the constraints is easy deployment. Ease of deployment will be explored later in this specification.

2.4 Main Features

The main features of this system are a donation subsystem and a scheduling (admin) subsystem. These subsystems will have a frontend and backend.

2.4.1 Backend

Both backends will have a REST API. A REST API makes it easy to access from multiple applications and easy to reason about. The donation subsystem will include:

- Registration/Login
- Donation without a trial
- Donate and schedule a trial
- Request a trial cancellation
- Get user feedback

The administration/scheduling subsystem will include:

- Confirm a scheduled trial
- Deny a scheduled trial
- Reschedule a trial (and notify user)
- Create a new trial (from phone or in person donation form)

2.4.2 Frontends

The frontends will come in web, iOS and Android clients. The web client will satisfy all features, have additional information and not require login. The iOS and Android clients will support all features but be more minimal and require login. The web admin client will support all the features of the backend scheduling subsystem.

3 Operational Description

This section includes a description of the development and production environments, a system model, and a user interaction subsection. It is important to flesh out a proper development environment to ensure thorough testing and ease of deployment.

3.1 Development and Production Environments

This system will operate in all popular web browsers, with a focus on recent versions of Chrome, Firefox, Safari, Microsoft Edge and Internet Explorer. It will also be available as a native application for iOS and Android. It will be targeting students of The University of Victoria, and as such, will generally be accessed via a modern laptop or tablet. There is a possibility that users will also access the website via their mobile phones but it will be suggested to use the native application. Due to the various different screen resolutions and sizes, we will ensure that components are dynamically resizable and fit for all scenarios whether it is for a vertical display for a mobile phone/tablet, or for typical desktop/laptop horizontal displays.

A suite of development tools will be used to develop the system. Git will be used for version control, along with GitHub for code review and project hosting. In order to ensure that new changes to the code do not break existing functionality, we have chosen to use TravisCI, a continuous integration system that can easily sync with GitHub. Finally, to manage ongoing tasks, bugs, project scheduling and organization, we will use Trello.

There will be a gated version of the REST API so that developers can access the development version of the backend. The frontend projects will use setup scripts so that the project can be setup with one script and run with one click. If this limits choice of development platform, that is a worthwhile sacrifice. The iOS application will be developed on a mac.

The backend will have a suite of tests that are always watched by a TravisCI job. This job will be responsible for pushing the latest dev build to the development server and running the new instance. If any tests fail the new build will not be pushed to the server. It is important to have all these steps scriptable to ensure ease of deployment to production. Less human steps involved means less potential for error.

3.2 System Model

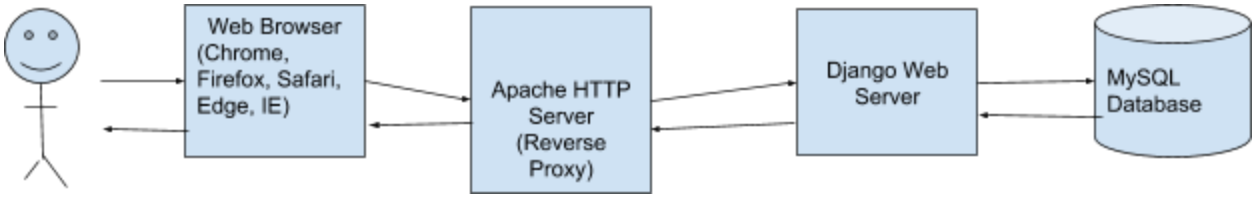

Figure 1: High-level system architecture

The system will be composed of four parts (see Figure 1):

- the website running on the user's web browser,
- an HTTP server,
- a web server,
- and a relational database.

The user will load the HTML/CSS/JavaScript website through their web browser. When a form is submitted, or a new page is requested, the user's web browser will send an HTTP request to the HTTP server. The HTTP server will analyze the request and forward valid requests to the web server. The web server will handle the request, reading or altering the data in the database. The web server will then send a response to the HTTP server, which will forward the response to the user's browser and update the browser's display accordingly.

The only hardware required to build and maintain the product will be a personal computer such as a laptop. A server rack will be required for hosting the finished product.

3.3 User Interaction

One of the most important aspects of this web application is how the users interact with it. The website has to be as intuitive as possible for the user and must have a donation subsystem that intuitively guides the user through each step of the online donation process and gives options for other donation methods such as in person or by phone. The two main type of users include the regular users (users that are in the website for more Order of PI information and possibly intending to donate), and Admin users (ESS and Monks of PI that use the web application to track the donations in real-time).

3.3.1 Regular Users

Regular users will have the ability to register to the system by providing their details in a form display, users with already existing accounts will be able to login upon entrance of the website by providing their username and passwords. They will be able to see each donation transaction from the past including the timestamp and the exact amount of donation. They will also be able to make donations using their username or to make the donation anonymously. Users will be able to "cancel" trials as well. This would send a cancel request to an administrator.

3.3.2 Admin Users

Admin users will be given the ability to see all donation transactions that have gone through the system through a list ordered by the date and time of the transaction. They will also see the name of the user that made the donation, or if the user has chosen to donate anonymously then the word "anonymous" will appear where the name should be accompanied by the amount of money in CAD that the user has donated. Administrators will be able to confirm or deny trials as well as accept or deny requests to cancel or reschedule trials. Admin users can also create new trials to input phone/in person donation requests.

4 Requirements

This section lays out the system's features and attempts to map those features to requirements. The concrete functional and non-functional requirements will be used to plan epic/stories and drive agile sprints. System testing will map directly to requirements to ensure all requirements are met.

4.1 System Features

The following are the system features that dictate the functionality of the system, and the functional requirements pertaining to each feature.

4.1.1 Time Allocation and Conflicts

The system shall provide functionality for users to schedule trials.

4.1.1.1 Functional Requirements

REQ-1: The system shall recognize and prevent double bookings
REQ-2: The system shall display available times for trials
REQ-3: The system shall not deny a user from moving forward with an available time slot
REQ-4: The system shall resolve all potential conflicts, not requiring intervention from the user
REQ-5: The system shall introduce extra constraints when a user confirms that the trial being scheduled is expected to occur during a lecture
REQ-6: The system shall assign to each scheduled trial a unique identifier to facilitate payment tracking

4.1.2 Payment Tracking

The system shall provide a payment method to accept payments directly from the website.

4.1.2.1 Functional Requirements

REQ-1: The system shall provide a payment portal such that users are able to pay for trials directly on the website
REQ-2: Each transaction shall be inextricably linked to a single trial via the trial's unique identifier assigned by the scheduling system
REQ-3:

4.1.3 Personnel Management

The system shall provide a personnel management system for the scheduling of personnel.

4.1.3.1 Functional Requirements

REQ-1: The system shall display scheduling information on all personnel, indicating when each member is available for duty

REQ-2: The system shall allow personnel to change their availability schedule
REQ-3: The system shall not allow personnel to change the availability schedule of other personnel (excepting for administrator users, who can view and change the availability schedule for all personnel)
REQ-4: The system shall provide a superuser (administrator) account, which has permissions to view and change scheduling information for all personnel

4.1.4 Warrant Printing

The system shall provide functionality for personnel to print paper warrants for trials.

4.1.4.1 Functional Requirements

REQ-1: The system shall display all currently pending (scheduled for some time in the future) trials
REQ-2: The system shall allow personnel to select any currently pending trial and print all pertinent information (these data fields are subject to change and not listed here)

4.2 Non-Functional Requirements

The following are a number of non functional requirements relevant to the application

4.2.1 Scalability

The application will be able to handle high traffic expected during the month of march, when the majority of users will be accessing the website.

4.2.2 Security

User payment information, and personal information must be encrypted before being stored. Users will not be able to access any information relevant to other users on the site. As the website does not involve any authentication or login, the payee will be provided with a receipt, including a unique identifier. This identifier will link the user to the payment.

4.2.3 Maintainability

Code will be clear and well documented so that changes can be easily made if they are needed. Regression testing and test driven development will be used to ensure that functionality is maintained as the codebase changes, as well as ensuring high test coverage. This will greatly improve the maintainability of the code in the long run.

4.2.4 Usability

Users will be able to navigate the application with little to no technical knowledge. Application will have clear titles, links and information regarding payments from the main page. It should be fast and intuitive to go through the entire process of making a donation.

4.2.5 Performance

Response times will be under 1 second at least 99% of the time, even during the month of March.

4.2.6 Availability

The application will be accessible 24 hours a day. If system failure should occur, all user data will be recoverable and time to recovery will be brief. When updates are needed, users will be notified in advance and downtime will be specified.

5 Feasibility

5.1 Design

Drafts of design concepts have been created and can be seen below in figure 2. Creating the mockups of the application help to understand the feasibility of the design, and if the outlined plans are actually possible and make sense. Through building a complete set of mockups we can determine if the app has logical flow and if the various components we drafted up actually make sense when combined. The figures below show a view of a generic user's home screen, then what it appears like when placing a trial, and the third one shows a list of trials likely from an administrative perspective.

Figure 2: Design sketches

5.2 Test Plan

The testing plan in place for feasibility is straightforward. We will be testing each action a user could take in our system, and ensuring that the desired outputs are achieved for each possible input. Our tests will then take a look at making sure our end product is able to work in the destination environment by executing load testing and seeing how it will perform once many requests are happening at once and the database begins to increase in size. The tests below constitute all the use cases for the system.

5.3 Client Perspective

- Register for an account, and be able to change the password if desired, as well as recover the account via e-mail address if password forgotten.
- Log in correctly, as well as with incorrect credentials to ensure errors are handled appropriately.
- Make a donation via payment processor without a trial.
- Make a donation via payment processor and schedule a trial by filling out the request form
- Ensure the request form properly sanitizes data and displays appropriate messages on invalid inputs.
- Cancel the trial, ensure user is prompted for optional refund.
- Ensure a feedback form is displayed so that users can let us know about any needs.
- Validate form feedback is stored and emailed out to PI admins.
- View list of trials I've scheduled under my account, old and current.
- See if my current trials have been confirmed by the admins.

5.4 Client Pie Recipient Perspective

- View list of current trials against my name.
- Allow myself to counter with a higher donation value through the payment processor.
- Show the trial request form for the Pi redirect if countered.

5.5 Admin Perspective

- View list of all trial requests .
- Approve or deny a trial request, and fill out a message to deliver to the client explaining why it was denied.
- Reschedule a trial and ensure user is notified of the change.
- Manually create an automatically approved trial from a phone or in person donation form.

6 Implementation

The implementation section outlines a feature complete version of the application. The following diagrams communicating critical details of the application. Figure 3 shows a view of the backend system. Figure 4 is a sample of feature complete mockups. The mockups will be used to derive native iOS and Android applications.

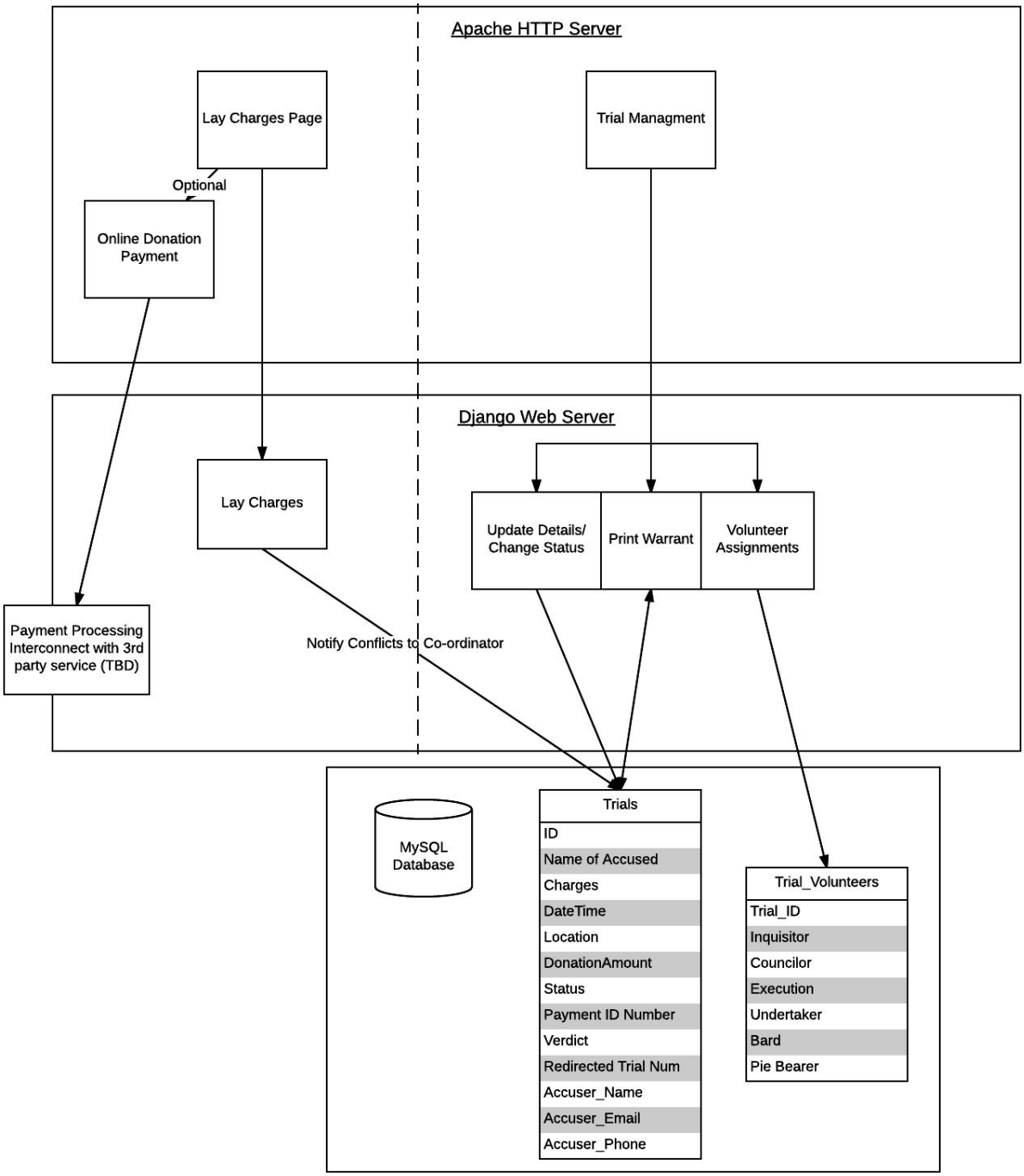

Figure 3: System and Database Diagram

Figure 4: Design Sketches