Order of PI: *Webscale*

Final Report

December 16, 2016

0xf Solutions ltd.

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

Table of Contents

1 Introduction

This document contains the build specifications for the final build of the Order of Pi's website created by *0xf Solutions ltd*. The following sections will give a design overview and how the final build met all the design requirements and mitigated any unforeseen behaviours. There will be a cost breakdown that will give a detailed analysis of the project's costs using the Constructive Cost Model. Creativity with the build will be described how the final build gave a modern look but still appealed to the required design of the monks of Pi. Accurate representations of the interface will be included to give a look at the styling and interface of both the user and administrative views. Qualitative test will be done on the front-end using the Capybara acceptance testing framework, and Quantitative tests will be done using both user-based authentication testing and Python unit-tests to properly cover the necessary scope of the code.

The final build is an exemplary improvement over the prior system's design both visually through the User Interface and responsively through the use of the Django framework. The requirements elicitation process was performed impeccably and there were no reasons as to have further elicitations sessions and thus the final product was able to be created quickly. As the proper tests have been put in place, the expectation for maintenance is minimal and the product will be able to function with no modifications seemingly indefinitely without taking into account security patches to the required frameworks. When the system is fully deployed for the Order, they should see an increase in volunteer rate, charity profits, and a decrease in management time for the fundraiser.

2 Design Overview

The ultimate design outlined in this document is the product of many iterations and borne of contributions from each team member. The agile-inspired process of project planning and requirement elicitation enabled *0xf Solutions* to refine the design and ensure sufficient requirement coverage. Stakeholder feedback received during the requirements elicitation and prototyping phases strongly influenced the final designed presented in this report.

2.1 Meeting Requirements

Functional and non-functional requirements have been formally specified in the submitted requirements specification document [1]. Each of the requirements specified is met or exceeded by the design of system components.

In order to meet requirements regarding scheduling trials, a system component was designed to provide an interface and backend for viewing and scheduling upcoming trials. This interface provides users a facility to book a trial in the future, and provides administrator users a facility to view all upcoming trials. In order to further meet requirements, the system component is constrained such that

- available times for trials are displayed
- users can book any time slot that displays as available
- double bookings are prevented, with potential conflicts transparent to the user
- some extra constraints are introduced for trials that are entered as occurring during a lecture, and
- each trial is assigned a unique identifier to be used by the system component charged with tracking payment.

To meet the requirements elicited regarding payment tracking, a system component was designed to provide an interface and backend for entering and tracking payment information. This interface provides administrators with a facility to track payments for each trial scheduled in the system, and provides users with a means to pay directly for trials using the interface. This component will be constrained such that

- each trial will be associated with a unique identifier, which is used to track payment for that trial, and
- each trial not paid for directly using the web app is tracked by administrators until payment is received.

Requirements regarding personnel management are met by a system component designed to provide an interface and backend for scheduling Order of Pi personnel. The interface provides Order of Pi members a means to enter their scheduling information. The interface further provides system administrators with a facility to view and alter all scheduling information in the system. This system component will be constrained such that

- Order of Pi members who are not system administrators cannot change scheduling information for other users, and
- Order of Pi members who are not system administrators cannot view scheduling information for other users.

To meet requirements regarding warrant printing, a system component was designed to provide an interface for printing paper warrants for trials. This interface provides Order of Pi members a means to select a trial from all pending trials and print a paper warrant for that trial, with most or all of the warrant details written in automatically.

2.2 Mitigating Emergent Behaviour

An important consideration in the implementation of any software system is obtaining consistent, deterministic behaviour. 0xf Solutions ltd. has produced a design that adheres to several principles that mitigate emergent behaviour in software systems.

Firstly, coupling between system components has been kept to a minimum, and where possible, system components are not coupled in any way. This ensures that erroneous behaviour exhibited by one system component does not affect the operation of any other component(s) in unexpected ways. Some coupling between system components is obviously necessary, as some component subsystems are common to several system components (i.e., the data store).

Secondly, an emphasis on testing will identify any emergent behaviours so that they can be rectified before being released with the software. 0xf Solutions has devised a thorough and rigorous test plan (see § 6) that should identify any emergent behaviour introduced in the current iteration such that system behaviour is consistent across iterations.

Finally, emergent behaviour will be mitigated by keeping the project codebase small and easy to maintain. By ensuring that the software does as little as is necessary to accomplish its tasks, emergent behaviour due to complexity is mitigated. Keeping the codebase maintainable further improves the consistency of system behaviour by enabling developers to easily rectify any unwanted behaviour or interactions between components.

3 Process Economics

To estimate the cost of the project, we used the Constructive Cost Model (COCOMO). It is a model often used for estimating the effort, cost and schedule for software projects [2]. This model uses an estimate of program size based on lines of code to determine the cost of the project. The following section shows an in depth breakdown of how our cost estimate was determined.

3.1 Cost Estimation

The following calculation gives us an estimate of overall effort, in man months, required to complete the project:

$E = a_i(KLoC)^{(bi)}(EAF)$

Where a_i and b_i are based on the size of the team. We determined our team to be an organic team, meaning that it is small, but has good experience. This gives us,

$a_i = 3.2$
$b_i = 1.05$

We also estimated thousands of lines of code (KLoC) to be 10

The relevant cost drivers have been assessed, and given an associated importance value, which correspond to an effort multiplier. These effort multipliers are then multiplied to get an effort adjustment factor (EAF).

Product attributes		
Required software reliability	1.15	High
Size of application database	0.94	Low
Complexity of the product	1.00	Nominal
Hardware attributes		
Run-time performance constraints	1.30	Very High
Memory constraints	1.00	Nominal
Volatility of the virtual machine environment	1.00	Nominal

Required turnabout time	1.00	Nominal
Personnel attributes		
Analyst capability	1.00	Nominal
Applications experience	1.00	Nominal
Software engineer capability	0.86	High
Virtual machine experience	1.00	Nominal
Programming language experience	0.95	High
Project attributes		
Application of software engineering methods	0.91	High
Use of software tools	0.91	High
Required development schedule	1.00	Nominal

Taking the product of the effort multipliers, we get an EAC of 0.95.

This gives us a total cost of 34 (in man months). Given that our group consists of 8 individuals, we estimate that we can implement the project in 4.3 months.

4 Creativity

The new design for the web application for the Order of Pi focuses on improved aesthetics as compared to the current system and added functionality for accepting and tracking donations . The new layout for the website better highlights pertinent information (e.g., important dates of the Order of Pi, and the description of the goals of the Monks of Pi) using the left center of the screen, which is the area of the screen where the eye of a user is first attracted [3]. Also on the main page, the selection tabs at the top of the screen with the different links (*Home, Rules, About, Lay Charges, Donate*) are strategically placed for a user's eye to find easily. The main page also features a geometric pattern with a friendly yellow/orange background that contrasts well with a white typeface. The rest of the website's pages make use of a theme that is based on this yellow/orange background from the main web page.

For the admin interface page for tracking and accepting donations, the distinctive design has donations organized in individual blocks containing all the information necessary about the donation including the amount, date, and the contact information of the person donating. Buttons for accepting or denying donations are conspicuous within the blocks. The design team found that this was a unique method and a simplest and effective way to see all pending donations.

5 User Interface Prototype

Accurate sketches to help understand the main components of the application can be seen below in figures 1 and 2. Figure 1 illustrates the homepage of the app. Along the top menu bar is the title, as well as a menu which links to the various important pages. There are links to come back home, view the rules, view the history of the Order of Pi, a donation button, and a link to the lay charge form. The large splash area in the middle displays information about the upcoming week of Order of Pi, and provides a quick intro to new users who are not familiar with what it is about. To the right of that is a miniature version of the *lay charges* form which will lead to the main form once filled out and submitted.

Figure 2 outlines the administrative interface of the app. Below the header is a list of links to tasks available to administrators such as user management, view donations, confirmation, denial or rescheduling of trials, and a form to manually create new trials from in-person or phone requests. As shown in figure 2, the admin interface is hidden from the home screen such that users are unable to find the admin login page.

The remaining pages (these pages are not yet prototyped) are all very similar in structure to the pending trials page (see figure 2). There is a header, followed by the respective content for that page. The dimensions of various components are not set explicitly as they vary depending on the users device and the size of their viewing screen. As the viewing window of the device changes, the app scales to fit appropriately and content is scaled down into increasingly fewer columns in order to be easily viewed from a mobile device.

The advantage to modular UI design and creating reusable components is that it is quick to add new pages using existing widgets and components. The additional pages will be quickly prototyped, user validated, and introduced into production.

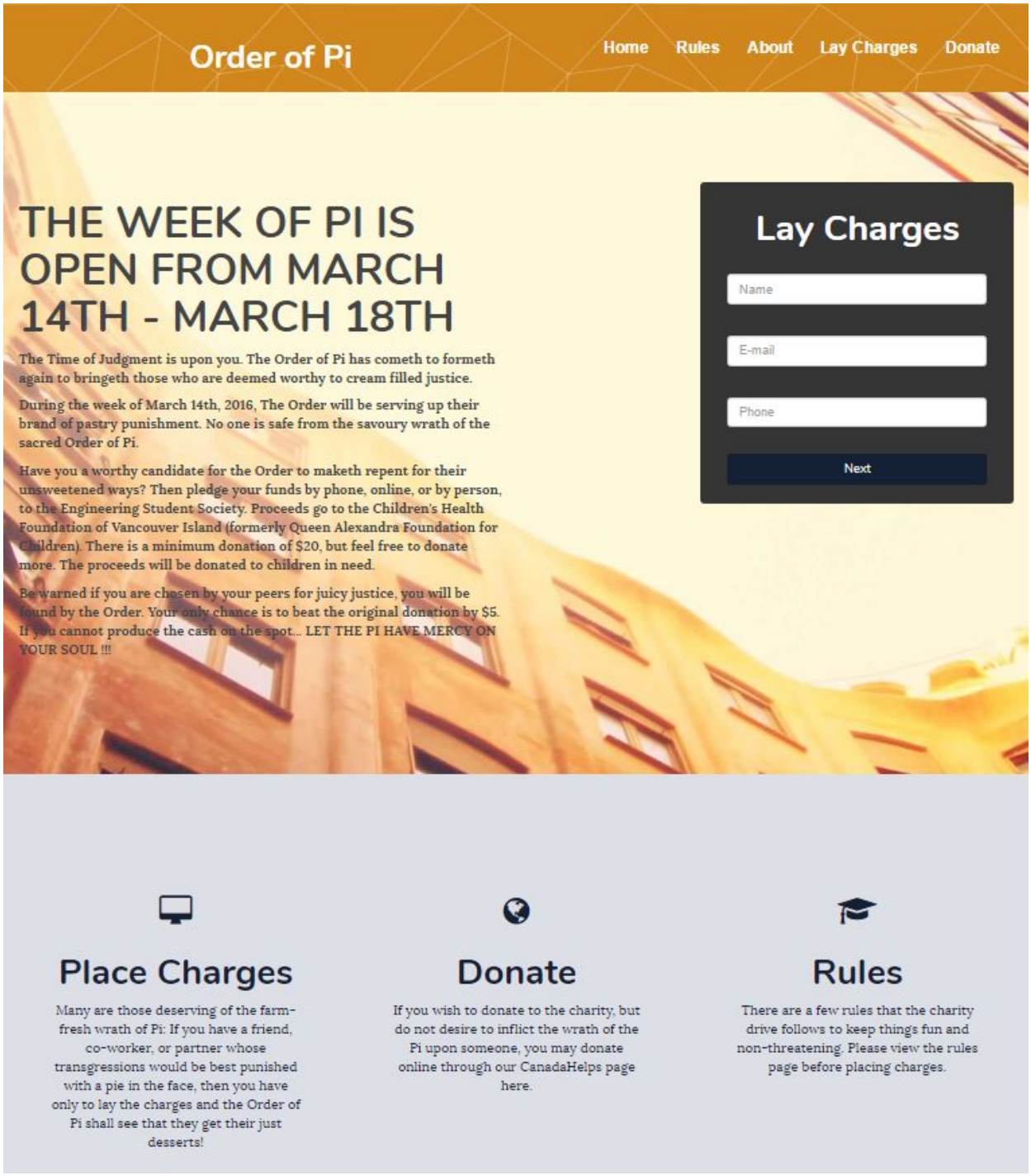

Figure 1: Order of Pi Home Layout

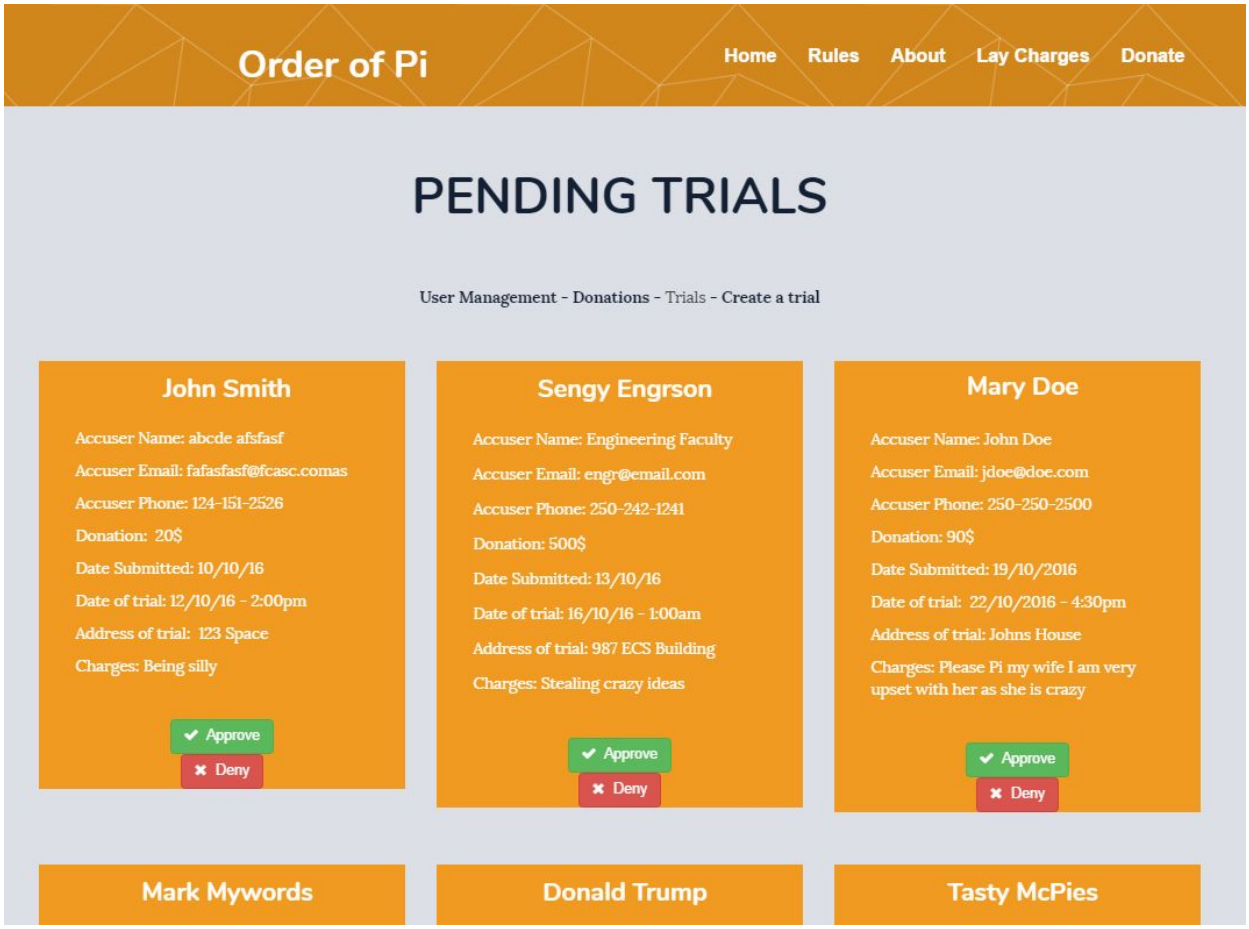

Figure 2: Order of Pi Admin Interface

6 Test Plan

The testing plan includes both qualitative and quantitative testing. Unit tests will be written for every piece of code that was committed. It will be each developer's responsibility to write tests for his code. The entire test suite will be run by the continuous integration (CI) service every time a new change is pushed to the GitHub git repository. Before a new set of changes can be merged into the master branch and become part of the production code, the new code will be required to pass all tests, both client-side and server-side. In addition, any client-facing features will be tested manually be the implementing developer to catch any bugs missed by the automated tests, and to ensure the features work as required.

6.1 Qualitative

The front end (client- or browser-side) code will be tested using the Capybara client-side acceptance testing framework. The client-side tests will use a JavaScript driver to automatically test feature functionality. For example, a test could be written to test that the login form is submittable and clickable by a user. These tests will primarily be used for regression testing: to ensure that existing functionality is not broken by new changes. In addition

6.2 Quantitative

Back end (server) code will be tested using the Python unittests library. Each function will be tested using black box unit testing. For example, unit tests will be written for each function that test that all types of input result in the expected output. Writing unit tests during development will allow development to proceed more quickly, since this allows developers to be be confident in the correctness of previously committed code. Furthermore, unit tests allow other developers to more easily understand the purpose of another developer's code, further increasing development speed.

In addition, integration tests will be implemented for the different types of web requests that the server will be expected to handle. For example, if the Django server receives an HTTP POST request to the "/login" URL, along with a username and password submitted as form data, integration tests would ensure that the appropriate response is returned by the Django application. In this case, either an HTTP "401 unauthorized" in the case that the provided username and password are not valid or HTTP "200 OK" if the credentials are valid and authorization succeeded. These integration tests will simulate different requests with different data and check that the given response is what is expected. The integration tests will ensure that all server code for a given action works together correctly.

Both unit and integration tests will serve to decrease development time and ensure correctness. In addition, both will also serve as regression tests that will ensure that any new changes to the code do not break existing assumptions or functionality

References

[1] 0xF Solutions ltd., “Order of Pi: Webscale Requirements Specification,” § 4, unpublished.

[2] B. Boehm, *Software Engineering Economics*, Upper Saddle River, NJ: Pearson/Prentice Hall, 1981.

[3] P. Laja, (2012). “10 Useful Findings About How People View Websites,” *ConversionXL*, 2012. [Online] Available at: https://conversionxl.com/10-useful-findings-about-how-people-view-websites/ [Accessed 17 Dec. 2016].