A web-based policy management tool to enable simple and robust management of end-user and server devices

William Stuart Oldham

University of Lincoln, School of Computer Science
19693813@students.lincoln.ac.uk

Abstract. A key part of any IT infrastructure is the system used to manage devices owned by the company. Such systems typically adopt a policy based approach where each policy is evaluated against each device and checked for compliance. Should a device not be compliant with the policies, a set of remedial actions are run to bring the device into compliance. Alongside knowing whether devices are compliant, knowing information about devices that are being managed allows IT professionals to work more efficiently. This project proposes a web-based management system used to manage and execute policies on client devices with minimal client device interaction.

Keywords: Software Development · Policy Management · Web GUI ·

Word Count:

1417 words

1 Introduction

Throughout the IT industry, a key part of a system administrators job is to manage a number of end-user client and server devices. While managing a device can consist of a number of tasks, one task is ensuring the device is compliant with policies that the system administrator puts in place. For large IT teams, there is typically a group of people designated with the task of monitoring compliance and modifying policies for the organisation. However, in smaller IT teams this is not always possible - especially if there is quite a high ratio of devices to IT staff.

This paper falls into the topic of "IT Automation" where a number of well-established tools exist. Some areas such tools fall down in and where this project is intended to target are: ease of installation and use, the type of push/pull architecture used and accessibility.

Pull architecture will be utilised where each client will request information from the server. As discussed in Martin-Flatin (1999), this operation can be more highly scalable that push architecture. Pull architecture is also particular advantageous for end-user devices as it can be used even when the device does not have a static IP address.

The interface in this project will be designed GUI-first where administrators will interact with the system using a web-based graphical user interface (GUI) as opposed to a command line interface (CLI). GUIs have the benefit that each task is less complex to complete which decreased the cognitive load an administrator has (Barrett et al. 2004, 393). Existing systems such as Ansible, SaltStack and Chef were designed initially with a CLI with the GUI being either an afterthought or a third-party add-on. The project proposed in this paper will be designed with the GUI being the primary focus so that the user experience is highly desirable.

A web-based interface is to be created that allows the user experience to not only be desirable, but also portable across multiple devices. As detailed in Weiss (2005), Weiss talks about the shift from computers to mobile devices. More than 15 years later, the movement to web applications is adopted by many large companies such as Microsoft, Apple, Google and others. With the interface for this project being web-based, an administrator can access the interface on any internet-connected device without installing any additional software. Having the interface being web-accessible means that system administrators are not restricted to working on laptops or desktop computers, but can also manage systems on a tablet device or even a mobile phone.

2 Aims and Objectives

The aim of this project is to produce a system which any system administrator would be able to easily manage devices with. The system will be administered through a graphical web-based interface designed using practices detailed in Haber & Bailey (2007).

The client operating system (OS) which shall be targeted is Ubuntu 20.04 LTS

There are 3 main components which will be produced as part of this project:

– Backend - The API server responsible for talking to the frontend and client.
– Frontend - The web interface which the administrators interact with to control the system.
– Client Agent - The program installed on each client device which will run any actions the administrator has entered.

Each of these components will communicate as part of collective group and will operate together to form one cohesive application.

A list of objectives are as follows:

2.1 Create an interactive web-based dashboard

– Use a reactive front-end framework with asynchronous requesting for the dashboard to allow operations to be done quickly without having to wait for pages to load.
– The dashboard will allow bulk operations to be carried out.
– The design of the dashboard should adhere to best practices in regards to user interface (UI) and user experience (UX) design.

2.2 Implement the policy management component

– The policy system should be
– All of the following policy types should be implemented on the server and the client:
  - Updating whole files
  - Setting specific configuration values
  - Package management (Using `apt` for the target of Ubuntu)
  - Running scripts
– Create at least 5 built-in policies that can be applied to a device

2.3 Implement a user management system

– Login & Logout – Session management
– Generating a secure default password

- Option to force the user to change the password on login
- Allow the user to edit their user details including name and password

2.4 Implement the client agent

The client should have all of the following functions as a minimum:

- Be able to be installed from a single command line instruction. For example: `wget http://policy-server.local/clients/linux-installer | sh`
- Request policies from the server at a defined interval. This interval is configurable from the web interface.
- Report hardware and software inventory back to the server.
- Heartbeat back to the server with current status – server will respond with either a blank response or a response indicating to the client that an early policy update should be ran – Allows server admins to force update clients without waiting for the policy request time to lapse.
- Run any policy actions which are determined as being uncompliant from the server's information.
- The client should be able to update itself without user intervention.

3 Academic Literature

Throughout the project, various literature will be studied to understand the methods currently employed in network management solutions and how to utilise best practices in the artefact. Martin-Flatin (1999) discusses the pros and cons of push and pull based management architecture and provides examples of interactions between server and client. Other literature such as Haber & Bailey (2007), Barrett et al. (2004) and Takayama & Kandogan (2006) will be used to inform the user interface and user experience design of the front-end web dashboard to ensure that it can be used efficiently by system administrators. Satti (2019) evaluates the security of configuring Linux systems and will help inform the security mechanisms which are implemented in the course of this project.

4 Project Plan and Risk Analysis

4.1 Project Timeline

The high level breakdown of the project is shown in fig. 1.

Using the high level timeline of the project, a more detailed task list was produced as shown in fig. 3.

4.2 Risk Analysis

The following risks analysed use the risk matrix as shown in fig. 2.

Risk: Running out of time

Initial Risk: L: 2 $\times$ I: 3 = High (6)

Due to the relative complexity of this project, some tasks could take longer than initially anticipated.

Mitigation: As part of the planning processes of each component, the minimum viable product (MVP) for that component will be considered. The MVP will then be used as a fallback if other components take longer than anticipated.

New Risk: L:1 $\times$ I:2 = Low (2)

Risk: Operating System Incompatibility

Initial Risk: L: 1 $\times$ I: 3 = Medium (3)

During the development of this project, it is possible that changes to the OS may cause components to not function correctly. This could mean that parts of the system would have to be changed swiftly to ensure a functioning program.

Mitigation: The 20.04 version of Ubuntu was selected as the target OS due to it being a long-term support (LTS) version. Due to its stability, the OS is highly unlikely to be changed or deprecated before the project is completed.

New Risk: L:1 $\times$ I:2 = Low (2)

Risk: Reliance on 3rd party libraries

Initial Risk: L: 2 $\times$ I: 2 = Medium (4)

This project will rely on a number of 3rd party libraries to ensure it can be completed in a timely manner. If a library was found to have a security risk or was to be deprecated, a potentially significant part of the project could be affected.

Mitigation:

New Risk: L:1 $\times$ I:2 = Low (2)

5 Acronyms

GUI	graphical user interface
CLI	command line interface
LTS	long-term support
UI	user interface
UX	user experience
OS	operating system
MVP	minimum viable product

6 Figures

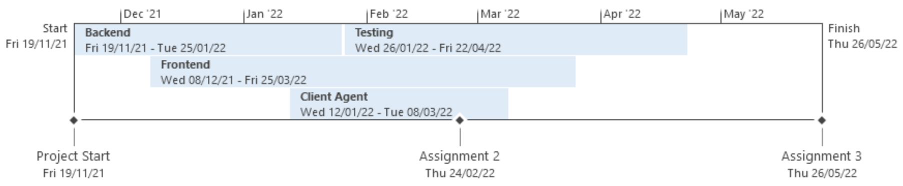
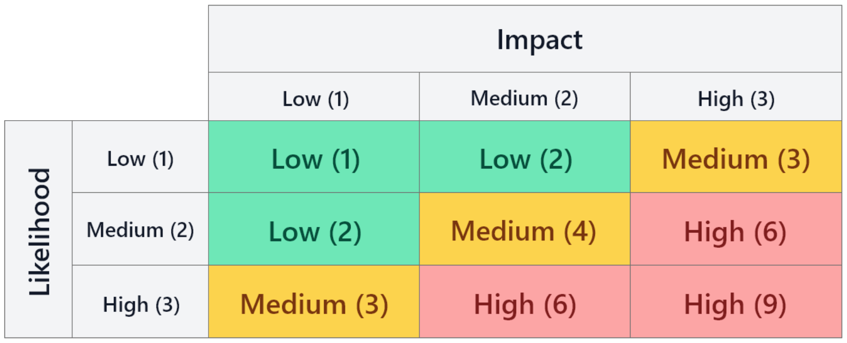
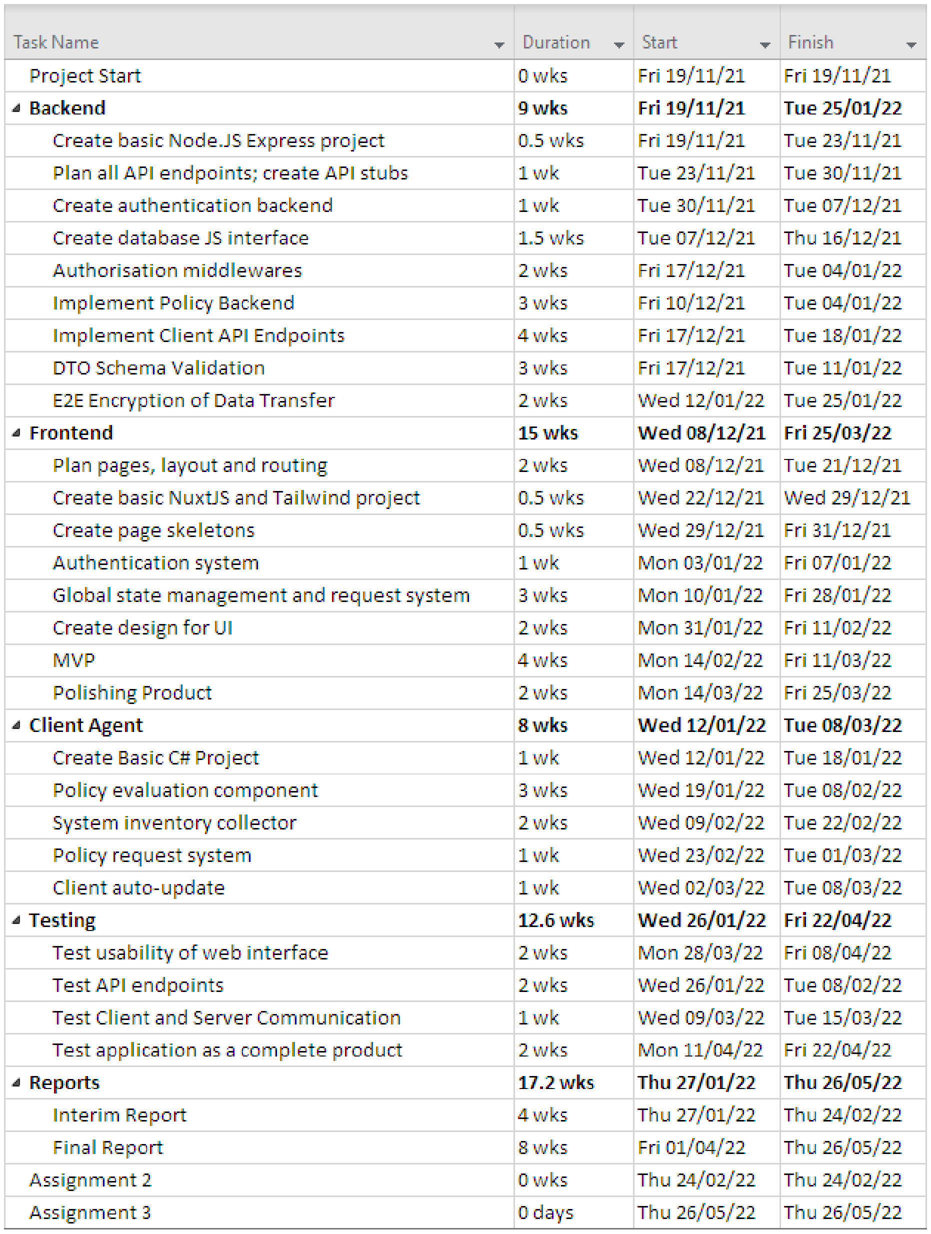


Fig. 1. Project timeline



		Impact		
		Low (1)	Medium (2)	High (3)
Likelihood	Low (1)	Low (1)	Low (2)	Medium (3)
	Medium (2)	Low (2)	Medium (4)	High (6)
	High (3)	Medium (3)	High (6)	High (9)

Fig. 2. Risk Matrix



Task Name	Duration	Start	Finish
Project Start	0 wks	Fri 19/11/21	Fri 19/11/21
Backend	**9 wks**	**Fri 19/11/21**	**Tue 25/01/22**
Create basic Node.JS Express project	0.5 wks	Fri 19/11/21	Tue 23/11/21
Plan all API endpoints; create API stubs	1 wk	Tue 23/11/21	Tue 30/11/21
Create authentication backend	1 wk	Tue 30/11/21	Tue 07/12/21
Create database JS interface	1.5 wks	Tue 07/12/21	Thu 16/12/21
Authorisation middlewares	2 wks	Fri 17/12/21	Tue 04/01/22
Implement Policy Backend	3 wks	Fri 10/12/21	Tue 04/01/22
Implement Client API Endpoints	4 wks	Fri 17/12/21	Tue 18/01/22
DTO Schema Validation	3 wks	Fri 17/12/21	Tue 11/01/22
E2E Encryption of Data Transfer	2 wks	Wed 12/01/22	Tue 25/01/22
Frontend	**15 wks**	**Wed 08/12/21**	**Fri 25/03/22**
Plan pages, layout and routing	2 wks	Wed 08/12/21	Tue 21/12/21
Create basic NuxtJS and Tailwind project	0.5 wks	Wed 22/12/21	Wed 29/12/21
Create page skeletons	0.5 wks	Wed 29/12/21	Fri 31/12/21
Authentication system	1 wk	Mon 03/01/22	Fri 07/01/22
Global state management and request system	3 wks	Mon 10/01/22	Fri 28/01/22
Create design for UI	2 wks	Mon 31/01/22	Fri 11/02/22
MVP	4 wks	Mon 14/02/22	Fri 11/03/22
Polishing Product	2 wks	Mon 14/03/22	Fri 25/03/22
Client Agent	**8 wks**	**Wed 12/01/22**	**Tue 08/03/22**
Create Basic C# Project	1 wk	Wed 12/01/22	Tue 18/01/22
Policy evaluation component	3 wks	Wed 19/01/22	Tue 08/02/22
System inventory collector	2 wks	Wed 09/02/22	Tue 22/02/22
Policy request system	1 wk	Wed 23/02/22	Tue 01/03/22
Client auto-update	1 wk	Wed 02/03/22	Tue 08/03/22
Testing	**12.6 wks**	**Wed 26/01/22**	**Fri 22/04/22**
Test usability of web interface	2 wks	Mon 28/03/22	Fri 08/04/22
Test API endpoints	2 wks	Wed 26/01/22	Tue 08/02/22
Test Client and Server Communication	1 wk	Wed 09/03/22	Tue 15/03/22
Test application as a complete product	2 wks	Mon 11/04/22	Fri 22/04/22
Reports	**17.2 wks**	**Thu 27/01/22**	**Thu 26/05/22**
Interim Report	4 wks	Thu 27/01/22	Thu 24/02/22
Final Report	8 wks	Fri 01/04/22	Thu 26/05/22
Assignment 2	0 wks	Thu 24/02/22	Thu 24/02/22
Assignment 3	0 days	Thu 26/05/22	Thu 26/05/22

Fig. 3. Project task list

Bibliography

Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A. & Prabaker, M. (2004), *Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices*, *in* 'Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work', CSCW '04, Association for Computing Machinery, p. 388–395. Available from https://doi.org/10.1145/1031607.1031672 [Accessed 7 Nov 2021].

Haber, E. M. & Bailey, J. (2007), *Design Guidelines for System Administration Tools Developed through Ethnographic Field Studies*, *in* 'Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology', CHIMIT '07, Association for Computing Machinery, p. 1–es. Available from https://doi.org/10.1145/1234772.1234774 [Accessed 7 Nov 2021].

Martin-Flatin, J.-P. (1999), *Push vs. pull in web-based network management*, *in* 'Integrated Network Management VI. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management.(Cat. No. 99EX302)', IEEE, pp. 3–18. Available from https://ieeexplore.ieee.org/abstract/document/770671 [Accessed 7 Nov 2021].

Satti, V. (2019), '*Secure Configuration and Management of Linux Systems using a Network Service Orchestrator.*', Master. Concordia University. Available from https://spectrum.library.concordia.ca/985519/ [Accessed 11 Nov 2021].

Takayama, L. & Kandogan, E. (2006), *Trust as an Underlying Factor of System Administrator Interface Choice*, CHI EA '06, Association for Computing Machinery, New York, NY, USA, p. 1391–1396. Available from https://doi.org/10.1145/1125451.1125708 [Accessed 9 Nov 2021].

Weiss, A. (2005), '*WebOS: Say Goodbye to Desktop Applications*', *NetWorker* **9**(4), 18–26. Available from https://doi.org/10.1145/1103940.1103941 [Accessed 7 Nov 2021].