NHS Ideas Portal

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

of

University College London

Department of Engineering

MSc Computer Science Project: COMP0073

Candidate Number: PGHX8

27th September 2021

Abstract

This project encompasses the design and development of an Ideas Portal that enables members from various organisations to brainstorm as a community. The aim is to deliver solutions to challenges facing the NHS in collaboration with the UCL Industry Exchange Network (IXN).

Additional goals included being able to learn and combine the Django and Django-REST frameworks to create a service capable of identifying and publishing practical user ideas. I adopted a Test-driven development approach (TDD) under an agile development methodology. In doing so, the project sought to apply modern design techniques and critically evaluate these in a practical scenario.

The site achieved its initial development goals as well as incorporating several features beyond the original project scope. I succeeded in broadening my understanding of Python and Django alongside HTML, CSS and JavaScript. The outcome of the project is an early iteration of a platform that will have a significant impact; serving thousands of users both internally and external to the NHS.

As an individual, I am proud of the final outcome and of the exploration I have had over the summer. I believe that I have applied my knowledge and understanding of key computer science principles acquired throughout the year, fundamentally, software development to the best of my ability.

Acknowledgements

I would like to thank my supervisor Dr. Yun Fu for his support throughout this project and his position as a point of referral and advice.

I would also like to thank my family and my friends for the encouragement and various forms of support and encouragement throughout the high and the lows.

Contents

1 Introduction **1**

1.1 Project Overview 1

1.2 Goals and aims 2

1.2.1 Project Goals 2

1.2.2 Personal Aspirations 2

1.3 Project Management 3

1.3.1 Agile Development 3

1.3.2 Test-Driven Development 3

1.3.3 Regression Testing and Defect Triage 4

1.3.4 Project Plan 4

1.4 Report Structure 6

2 Background **8**

2.1 Technologies 8

2.2 Related Project Review 9

2.3 Back-end Tools 10

2.3.1 Linode 10

2.3.2 Django Framework 10

2.3.3 Django-REST Framework 11
2.3.4 PostgreSQL 11
2.3.5 Nginx and Gunicorn 12
2.4 Front-end Tools 13
2.4.1 HTML and CSS 13
2.4.2 Javacsript 13

3 Requirements **15**
3.1 Requirements 15
3.1.1 Problem Statement 15
3.1.2 Requirements Gathering 16
3.1.3 Use Cases 20

4 System Design **22**
4.1 Outline 22
4.2 High-level Architecture 22
4.3 System data Flow 24
4.4 Site maps 25

5 Implementation **27**
5.1 Models 27
5.2 Organisation set-up 27
5.3 Ideation 29
5.3.1 Challenges 29
5.3.2 Polls 30
5.3.3 Ideas 30

5.3.4 PRIDAR and organisation-specific forms 31

5.4 Back-end 33

5.4.1 ORCHA API 33

5.4.2 Authentication 34

5.4.3 Category Search and Keywords 34

5.4.4 Idea Validation 35

5.4.5 Email Verification 38

5.4.6 Deployment 39

6 Testing **40**

6.1 Functional Testing 40

6.1.1 Manual Tests 40

6.1.2 Automated Selenium Tests 40

6.1.3 Regression Testing 41

6.1.4 Unit Testing 43

6.1.5 System Integration Testing 43

6.2 User Acceptance Testing 46

6.2.1 Results 47

7 Conclusion and Project Evaluation **48**

7.1 Achievements 48

7.2 Fulfillment of Personal Goals 49

7.3 Critical Evaluation 50

7.4 Future Work 51

7.5 Conclusion 52

Bibliography **53**

Appendices **55**

A User Manual **55**

A.1 Login Details 55

A.2 Sign-in and Registration Pages 57

A.3 Landing Pages 60

A.4 Components 62

A.5 Help Pages 63

A.6 Polls and Challenges 65

A.7 Ideas 68

A.8 Profile Page and Settings 73

A.9 Ideas Hub 74

A.10 Challenge Management 76

B Deployment Manual **78**

C Use Cases **80**

D Code Listing **89**

E Detailed Gantt Chart **105**

List of Figures

2.1 Technology stack 8

2.2 Related Technology Review 9

2.3 Nginx congiguration file 12

3.1 PRIDAR Scheme from CNWL 16

3.2 Standard NHS/Public User Use Case 21

4.1 High-level view of ideation process 23

4.2 System Data Flow Diagram 24

4.3 Standard User Site Map 25

4.4 Advanced User Site Map 26

5.1 Org-hub HTML code 28

5.2 Org-hub UI view 29

5.3 Submit challenge function 30

5.4 Like idea function 31

5.5 JSON Serializer for ideas in Django-REST framework 32

5.6 GET Request from API 32

5.7 Succesful PATCH request to update idea stage to 'under review' . . 33

5.8 Authentication of ORCHA API in Django 34

5.9 Access token used in GET request to retrieve application categories in ORCHA 35

5.10 Keyword generation and comparison against words users input into the form 36

5.11 GET request to extract relevant applications that could be similar or duplicates 36

5.12 API integration to idea forms in front-end 37

5.13 Link to app stores with more information and downloads 38

5.14 Link to app stores with more information and downloads 39

6.1 Automated Functional test case example (user registration) 42

6.2 Unit test cases 43

6.3 UAT Survey 46

A.1 Sign in page 57

A.2 Enter Portal Page 57

A.3 Registration 1 - Username and password 57

A.4 Registration 2 - Age group 58

A.5 Registration 3 - NHS staff member? 58

A.6 Registration 4 - Select your interests 58

A.7 Registration Successful 59

A.8 Verification email 59

A.9 Activation complete 59

A.10 Access Denied Page 60

A.11 Custom Landing Page 60

A.12 Custom Landing Page 2 61

A.13 Navigation bar for admins and challenge managers 62

A.14 Standard Navigation bar 62

A.15 Footer 62

A.16 Main Help Page 63

A.17 Admin Info Page 63

A.18 Portal Manager Info Page 64

A.19 Challenge Manager Info Page 64

A.20 NHS Staff Info Page 64

A.21 Public User Info Page 65

A.22 Monthly Challenge Archives 65

A.23 Challenge Feed and Latest polls 66

A.24 Pagination example 66

A.25 Poll Detail Page 67

A.26 Poll Detail Page 2 67

A.27 Comment view 68

A.28 Submit an idea Form 69

A.29 Standard Idea Criteria Form 69

A.30 Idea Submission Success Page 70

A.31 Search Blog Page 70

A.32 Idea Detail Page 71

A.33 Custom (PRIDAR) Idea Detail Page 71

A.34 Custom (PRIDAR) Idea Management Detail Page 72

A.35 Profile Page 73

A.36 Edit age form 73

A.37 Edit email form 74

A.38 Idea Hub search 74

A.39 Idea Hub search 75

A.40 Idea Lifecycle timeline 75

A.41 Idea Stage view 75

A.42 Pending Challenges view 76

A.43 Completed Challenges view 76

A.44 Pending Idea view 77

E.1 Detailed Gantt Chart 106

List of Tables

A.1 Login details 55

C.1 UC1 Register to a portal. 81

C.2 UC2 Submit a challenge 83

C.3 UC3 Submit an idea. 85

C.4 UC4 Review challenges and ideas. 86

C.5 UC5 Update Idea Stage. 88

Chapter 1

Introduction

1.1 Project Overview

The NHS has long needed a centralised space for its users internally and externally to raise grievances and work together to solve these [12]. Current pathways include Microsoft Teams and surveys but this is difficult for scaling up and here the immediate challenge can be recognised. Furthermore, these pathways to not offer functionality to track the progression of ideas through the NHS Trustnet development life-cycle and is not integrated into existing NHS service. NHS Trustnet is a compatible platform that offers visibility of solutions at a high-level. After meeting with senior members and administrators of the platform it was clear that they wanted to integrate the 'winning' ideas from my portal into their site through an API and wanted me to facilitate this through intelligent design.

The leading client for the portal was a branch of the NHS based in Wales called CNWL. After meetings in the early weeks they communicated their need for organisation-specific forms, namely 'PRIDAR' a schemea used by the organisation to measure the viability of ideas. PRIDAR and its application which will be explored in detail later during the implementation chapter.

This project is also heavily integrated with an external ORCHA REST API that compares new ideas for challenges against existing ones with the aim of creating a refined space. The project aims to educate and encourage learning in protected spaces via discussions surrounding topics ranging from mental health to job satisfaction. Research proves the effectiveness of this approach to resolving challenges; justifying the requirement for this project [23].

Given the nature of the project and its aims, ethical considerations have been factored in throughout design and implementation - ensuring that NHS and data-protection guidelines are followed. There is a clear emphasis on a dialogue and challenges/ideas rather than the users themselves as it was communicated from inception that anonymity is also important to the success of the platform.

1.2 Goals and aims

1.2.1 Project Goals

1.2.2 Personal Aspirations

My aims consisted of applying the skills and knowledge that I have acquired throughout the academic year in a practical situation. I wanted to fill the gaps in my knowledge by improving my understanding of business requirements and software development so that I could solely deliver an application from start to finish. The flexibility of this project allowed me space to build a plan that I believed would demonstrate my skill set whilst encouraging me to learn and explore areas within the computational sciences that I had not done before. I sought to immerse myself in the experience and work with Django and accompanying tools and technologies.

As I delved deeper into the project, I explored areas of software development that I had not experienced such as deployment and this made the experience very rewarding. I came to understand the importance of APIs in future applications and the effectiveness of the Django framework in building powerful, quick web

applications [21].

1.3 Project Management

1.3.1 Agile Development

I followed an agile approach comprising of iterative sprint cycles [11]. As part of this, weekly meetings were scheduled with clients and stakeholders from within various organisations in the NHS - primarily CNWL. This input was especially significant in the design; key to ensuring the application catered to the requirements of its users and performed at the highest standard [13]. My role can be described as that of a ”T-shaped individual” on the project; with knowledge in areas of the business requirements (Business Analyst) and software testing (QA Tester) but with a specialisation in web development (Web Developer) .

This is in line with the 3-amigos approach to agile development which identifies misunderstandings and confusion early and allows learning to happen sooner in the delivery of an increment of work[2][22]. The 3 amigos also builds a shared understanding about the intent of an increment of work; ensuring that all parties were bale to input into the design and implementation with aspects from their areas that would combine to produce a well-rounded and more complete product. Effectively my testers also included the users themselves, I specialised in the web development and a consultant for CNWL would moderate the meetings between myself and the organisation as well as additional stakeholder in other branches of the NHS.

1.3.2 Test-Driven Development

Since the initial meeting with the client, a significant emphasis was placed on the importance of the quality of the application’s key features. Hence, the value of rigorous testing was clear because testing is essential to quality assurance [10]. In context of this I made the decision to adopt a test-driven approach to development.

This approach required that I design test cases from my MOSCOW requirements and essentially debug throughout development rather than towards delivery. In doing so the application has been extensively tested from development to production and refined to minimise defects and bugs [14].

1.3.3 Regression Testing and Defect Triage

I used weekly sprint cycles; performing extensive regression testing and defect triage during weekends by triggering an automated script to validate key user journeys and features that could be affected by bugs throughout development. In doing so I was quickly able to identify, triage and implement fixes to bugs discovered in relation to key areas of functionality [20].

1.3.4 Project Plan

The following plan summarises the phases of project. Each week represents a sprint in accordance with agile principles [19], where I met twice with clients and once with my supervisor . A visual representation can be seen in the Gantt chart in the appendix (Figure D.1).

- 7th June - 27th June
  - Requirements Gathering and Analysis
  - UI Design
  - Development Setup (Database Design, GitHub Repo and Django project)
  - Learning Django and Required Learning
- 28th June - 11th July
  - Front-end Development
  - Functional Testing
  - Unit Testing

  - Regression testing

- 12th July - 15th July
  - Back-end Development
  - Performance testing
  - Regression testing
  - Integration Testing
  - E2E Testing

- 16th August - 22nd August
  - Deployment
  - Deployment and user manual writing
  - User Acceptance Testing

- 23th August - 29th August
  - Bug-fixing
  - Sanity testing
  - Regression testing

- 30th August - 13th September
  - Report Writing

1.4 Report Structure

The report is divided into seven chapters. The remaining chapters will adhere to following structure:

- Background:

  This chapter details the inception of the project and explains the purpose of the NHS Ideas Portal and its value. I will begin by describing the technology stack that I used to build the site, before evaluating the strengths and weaknesses of its competitors and finally explaining the application of the tools I would use in development and deployment.

- Requirements:

  This chapter will establish the goals of the project and describe the process behind gathering information from the client. The requirements will then be sorted in order of priority and use cases will be explored to contextualise the project.

- System Design:

  This chapter will explore the design process undertaken prior to implementation; delving into the system architecture and planned user journeys/experience.

- Implementation:

  The core chapter of the report will outline the functionality of the site and how it was delivered through the use of the tools mentioned. It will provide an in-depth and digestable summary of the development phase of the project and conclude with the outcomes of the deployed application.

- Testing:

  This chapter will give insight into the rigorous testing approach used throughout implementation in line with TDD and agile methodologies. The types of testing including user acceptance (UAT) and unit testing will be covered in detail.

- Conclusion and Project Evaluation:

  To conclude the report I will evaluate the project as a whole; assessing the successes as well as the short-comings. The report will close with a note on the potential of new features and a reflection upon what I have learnt and the value of this experience to me both as a computer scientist and an individual.

Chapter 2

Background

2.1 Technologies

The project back-end was configured with a Nginx server and used Gunicorn as a gateway interface HTTP server with a PostgreSQL database integrated. The Linux-based cloud tool, Linode was supplied by the client due to affordability and ease of integration with existing platforms. Using the tool I was able to host the site on a virtual private servers at the following IP address: 139.162.251.153.

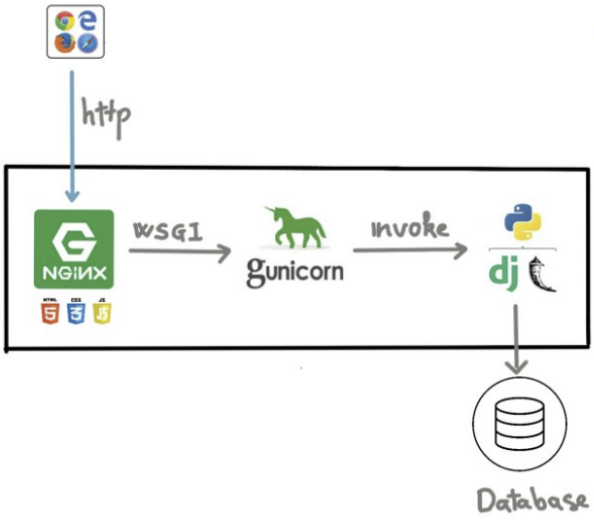

Figure 2.1: Technology stack

In addition to this, the suitability of Nginx to meet this product's usability requirements contributed to the decision to use this technology stack for the project. Given it is open to thousands of members of staff at the NHS as well as public users, Nginx's capabilities as an all-in-one load balancer, web server, API gateway, and reverse proxy meant that it suited the project's requirements almost perfectly.

2.2 Related Project Review

As part of the project design, I spent time evaluating related applications. The intention behind this was to highlight successful features as well as shortcomings that I could improve on within my design [17]. This was essential to defining MoSCoW requirements and influencing the design of the final prototype that I demonstrated to the client. I followed the recommendations of the client to review 3 applications in particular that they felt best replicated and captured their ambitions with the NHS Ideas Portal. The table below summarises the key features and weakness of said applications:

Name	Features	Weaknesses
Simply do	• Simple and streamlined registration process • Users can be segmented and affiliated with multiple groups • Social tools e.g. hashtags and likes for familiarity as well as allowing users to interact with each others • Customisable branding to allow the development of user interfaces that are tailored to groups	• The design is quite plain and may be boring to users
IDEAdrop	• Clean colourful bright UI • Short succinct forms for users to fill • Allows anonymous entries which can encourage higher engagement from users that want their identities protected • Challenge interactions are concise • Rewards engagement to motivate users to brainstorm effective solutions • Customisable workflow • Variable user permissions with different roles and capabilities within the site • Colourful and engaging UI that makes it stand out amongst competitors	• Web pages are very long and the UX could be more streamlines and seamless to make the site easier to use • Hard to locate and search for specific ideas
PRIDAR	• Clear and easy to understand with effective segmentation of idea criteria under the labels True or False	• Requires additional analysis to be performed on ideas to decide on the viability • Very simple and binary but in some scenarios challenges may require a more than a binary solution - this option also cannot be toggled

Figure 2.2: Related Technology Review

From completion of the related technology review I came to the conclusion that IDEAdrop was the closest competitor, nonetheless, I made the decision to factor elements of all technologies into my design. Commonalities in the ideation platforms that I sought to incorporate into my design include anonymous logins and

captivating user experiences. However, I aimed to surpass the current competitors through the API validation and inclusion of the PRIDAR schema to add to the overall complexity of the system. My ultimate goal was to incorporate key features that were well received by users whilst remaining under the guise of IXN/NHS their its design principles. As part of this requirement, I intended to use the NHS prototype kit during implementation to customise the site with traditional NHS formatting.

2.3 Back-end Tools

2.3.1 Linode

The decision to use Linode as the cloud hosting provider was based primarily on its exceptional performance and cost-effectiveness. The affordability factor lies in the corporate DNA of Linode, and as such, they were the first to introduce a flat pricing model for cloud computing. Additional advantages included the extensive documentation and ease of integration with similar NHS/IXN application in the cloud.

2.3.2 Django Framework

The python-based web framework, Django offered components that encourage fast and simple web-development[1]. This suited the needs of the project by allowing to incorporate as many of the features in Moscow requirements as possible This ultimately would enable me to develop the most complete application I could given the time constraint.

Django's built-in features including user authentication and effective database management made it the suitable framework for the project, given the higher prioritisation of functional features of UI by the client. In addition to this, the requirement for a public API set the precedent to use Django in implementation as a package with its corresponding REST framework covered in the next section. These capabilities would enable me to produce rapidly create web pages and seamlessly integrate

these with the PostgreSQL database .

2.3.3 Django-REST Framework

Django-REST framework (DRF) is a flexible and fully-featured toolkit with modular and customisable architecture that makes possible development of both simple, turn-key API endpoints and complicated REST constructs [7]. As mentioned in the introductory chapter it was necessary that the application have an accompanying web API so that winning ideas could be integrated within applications for the NHS, namely 'Trustnet' which was introduced in the opening chapter. DRF aims at building sophisticated web APIs, containing a wide set of out of the box features as well as being general is easy to use; a sensible choice for this project's implementation. A significant advantage of DRF is that it allows developer to define URL structure and not rely on an auto-generated one. Moreover, Django REST Framework includes a built-in API browser for testing out newly developed API [21]. This allowed me to effectively test the API endpoint to ensure that it met the requirements.Hence, it was used to create web APIs that would be used by Trustnet to update the stages of selected ideas throughout development.

2.3.4 PostgreSQL

PostgreSQL is an advanced, open source relational database that supports both SQL and JSON querying. It is a highly stable database management system.PostgreSQL seamlessly integrates with Django's model-based configuration and this made database management simple and effective.

PostgreSQL has a rich history for support of advanced data types, and supports a level of performance optimization that is common across its commercial database counterparts, like Oracle and SQL Server [11]. This meant the database has ability to handle the high usage expected from the portal.

2.3.5 Nginx and Gunicorn

Nginx was configured to act as a reverse proxy to handle client connections and serve the project depending on the client request. Nginx is built to offer low memory usage and high concurrency. Rather than creating new processes for each web request, Nginx uses an asynchronous, event-driven approach where requests are handled in a single thread [5].

```
server {
    listen 80;
    server_name 139.162.251.153;
    client_max_body_size 100M;
    sendfile on;
    tcp_nopush on;
    types_hash_max_size 2048;
# server_tokens off;

location /static/ {
    root /home/[redacted]/ideaportal-project;
}

location /media/ {
    root /home/[redacted]/ideaportal-project;
}

location / {
    proxy_pass http://139.162.251.153:8000;
    }
}
```

Figure 2.3: Nginx congiguration file

With Nginx, one master process can control multiple worker processes. Since Nginx is asynchronous, each request can be executed by the worker concurrently without blocking other requests.

Gunicorn was configured to translate client requests so that Django can handle them. Afterwards, I set up Nginx to act as a reverse proxy to handle client connections and serve the portal and its accompanying static and media files depending on the client's HTTP request.

2.4 Front-end Tools

2.4.1 HTML and CSS

Hypertext Markup Language (HTML) was used throughout web development because it allowed me to can create and structure sections, paragraphs, and links using elements, tags, and attributes. HTML was an effective tool in displaying web page elements in the project including the challenge/idea format and supporting media files.

CSS (cascading style sheet) was used to style the web elements and was essential to meeting the requirement using the NHS Prototype Kit mentioned in the previous chapter. This was accounted for a significant proportion of the UI design and was key to defining how users would be stimulated visually by the portal[17].

2.4.2 Javacsript

JavaScript is a text-based programming that was used in support of HTML and CSS to make certain web pages interactive, namely the challenge/idea review pages. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user. Incorporating JavaScript improves the user experience of the web page by converting it from a static page into an interactive one and contributing the an immersive UI/UX experience.

In spite of the design limitations that the NHS Prototype kit placed upon the portal

- I sought to manoeuvre around this where I could to avoid delivering a bland and boring application. In order to maximise responses and engagement for our users the value of this cannot be undermined. Hence, JavaScript was useful for confirmation and conditional rendering of web pages depending on what information users had entered into idea forms.

Chapter 3

Requirements

3.1 Requirements

3.1.1 Problem Statement

The NHS faces several challenges each day, from systemic issues to individual queries. There has been a persistent difficulty in tracking these issues because of the volume and absence of a centralised area for individuals to raise their problems. Currently, users raise issues via several avenues, for example, reporting directly to managers and conversations in MS Teams channels meaning that challenges are submitted in different formats and that there is no way of comparing them against a baseline.

Example PRIDAR Report for a Pathway

AI RISK STATUS

OK
UNKNOWN
OF CONCERN
AT RISK

NOT APPLICABLE NA

	DESIGN		DEPLOYMENT		ASSURANCE						ORGANISATION PREPARED					
AI Solution	User Led	In Sandbox	Released & Supported	Open Source Partnership	NICE Teir 1 + DTAC Evidence in Place	NICE Tier 2+ DTAC Evidence in Place	Risk & Mitigations are Public	Safety Officer Stated	CE mark/DCB Register	ISO Supplier	User KPIs in an AI Pathway are defined	User to Board Approval Obtained	Cost of Dev & Support Agreed	IP agreement in place	IG Agreements in Place	Data & model agreed
AI System 1	Yes	Yes	NA	Yes	Yes	NA	Yes	Yes	NA	NA	No	Yes	Yes	Yes	Yes	Yes
AI System 2	Yes	Yes	NA	Yes	Yes	NA	Yes	Yes	NA	NA	No	No	No	Yes	Yes	Yes
AI System 3	Yes	Yes	NA	No	NA	No	No	No	No	Yes	No	No	No	?	?	Yes
AI System 4	Yes	No	Yes	No	NA	No	No	No	No	Yes	No	No	No	Yes	Yes	Yes
AI System 5	Yes	Yes	NA	Yes	NA	No	No	No	No	No	No	No	No	Yes	Yes	Yes
AI System 6	Yes	No	Yes	No	NA	?	?	?	?	?	?	?	?	?	?	?
AI System 7	Yes	No	Yes	No	NA	No	No	No	No	Yes	No	No	No	No	No	No
AI System 8	Yes	Yes	NA	Yes	NA	No	Yes	Yes	No	No	Mo	No	No	Yes	Yes	Yes
AI System 9	Yes	Yes	NA	Yes	Yes	Yes	Yes	No	Yes	Yes	Mo	No	No	Yes	Yes	Yes
AI System 10	Yes	No	Yes	No	NA	?	?	?	?	?	No	No	No	No	No	No

Figure 3.1: PRIDAR Scheme from CNWL

This portal is intended to offer resolve this issue by offering that baseline against which ideas can be evaluated. The portal is intended for use primarily by CNWL - with potential for growth and use by more communities facing similar difficulties. Criteria for evaluating ideas should be defined on an organisation-wide scale so that communities have the ability to define and compare ideas against their own specified criteria.

3.1.2 Requirements Gathering

I held two weekly meetings, the first with the client from CNWL and the second with various stakeholders from organisations technical personnel in the NHS. The purpose of this was to discuss ongoing development of the portal and to encourage influence over the design process - making it highly user-focused [8]. Typically, an ambassador from each organisation (CNWL, Trustnet etc.) would attend each meeting to offer feedback and expectations from the system once complete. Each ambassador also explained how exactly the system would be used in their organisation and the different purposes further evidenced the importance of flexibility with regards to admins and portal managers being able to control activity within their portals.

It was imperative that the platform was stylized with the classic NHS design to

cater to its users that speak over 50+ languages and range of ages - highlighting the importance of ease-of-use [9]. Moreover, to distinguish this application from other ideation platforms, the client requested that the platform validates incoming ideas against existing ones to prevent duplicates, thereby refining the submissions to produce the most effective solutions.

On the basis of our regular conversations requirements adapted over time, however the final list of requirements was defined:

The ideation portal must:

- Functional Requirements
  - The portal shall enable all users to respond to challenges that have been posted with their own ideas
  - The portal shall allow users to view their own profile and idea/challenge history
  - The portal shall enable users to register and be authenticated as various user types (admin, public and nhs staff)
  - The portal shall allow users to log in and view data from earlier sessions
  - The portal shall allow users to post challenges depending on their permissions (defined by user type)
  - The portal shall allow admins and managers to view challenge statistics
  - The portal shall accept feedback on challenges (upvotes, downvotes etc.) so that the popularity of ideas can be gauged
  - The portal shall abide by data protection rules and GDPR (do not recommend direct solutions) (make specific)(delete)
  - The portal shall include a way to compare user feedback against existing responses and ideas (API) to validate that it is a new idea

  - The portal shall grant CRUD permission depending on the selected user type
  - The portal shall assess PRIDAR criteria for each challenge in order to determine the winning idea

The ideation portal should:

- Functional Requirements
  - The portal shall facilitate 2 further user types (portal manager and challenge manager)
  - The portal shall enable users to view the profiles and idea/challenge history of other users
  - The portal shall allow users to post responses to challenges anonymously
- Non-Functional Requirements
  - The portal shall have a unique and engaging UI/UX to stand out amongst competitors
  - The forms for challenge/idea submission shall be shall be compact and categorizable
  - The portal shall contain visual cues that show the popularity/interactions with a challenge

The ideation portal could:

- Functional Requirements
  - The portal shall have limitations around how many responses to a certain challenge or how many challenges a user can post within a timeframe to prevent spam

  - The portal shall assess a range of criteria to determine whether an idea is the winner (beyond PRIDAR)
  - The portal shall include more polling in other areas (User feedback and rating of the site)
  - The portal shall allow users to login by scanning a QR code
  - The portal shall notify users via push notifications, e.g likes and comments on their challenges (move to functional)

- Non-Functional Requirements
  - The portal shall allow users to have experience(XP) or a credibility rating to show the success of their previous ideas
  - The portal shall provide accessibility features for impaired users

The ideation portal won't have:

- The portal shall not use special effects instead remain simple and accessible
- The portal shall give direct suggestions of solutions to user challenges

The complexity of the requirements demanded that I create 5 user types with various roles and permissions within the system. The roles were divided in hierarchical order from the most senior to the least senior

- Admins
- Portal Managers
- Challenge Managers
- NHS Staff
- Public

3.1.3 Use Cases

The typical user journey would involve raising a challenge under the banner of a specific organisation or publicly. After submitting the challenge an admin or challenge manager reviews the 'ticket' and has responsibility over challenge management through its lifecycle. If they are content with the details of the challenge they can start a poll by navigating to the management area of the portal and setting a start and end date. In doing so, the idea is then published on the site and is visible by all members of the portal or all users if the 'make public' option is selected.

Users that can view the challenge then have the ability to suggest ideas, given that it is not a duplicate or highly similar to an existing idea for the challenge at hand. Organisations have the ability to define which criteria to use as a baseline for the relative comparison of ideas. This allows them to identify key pieces of information that change depending on the organisation. The ideas are then submitted and other users are able to interact - leaving comments and liking the ideas. Once the poll's end date is reached the idea with the most likes are then selected as 'winner' and its status updates automatically before being pushed to the local API. Trustnet is then able to update the status of the chosen ideas from 'open' to 'delivered' and this is reflected in the Ideas' hub timeline.

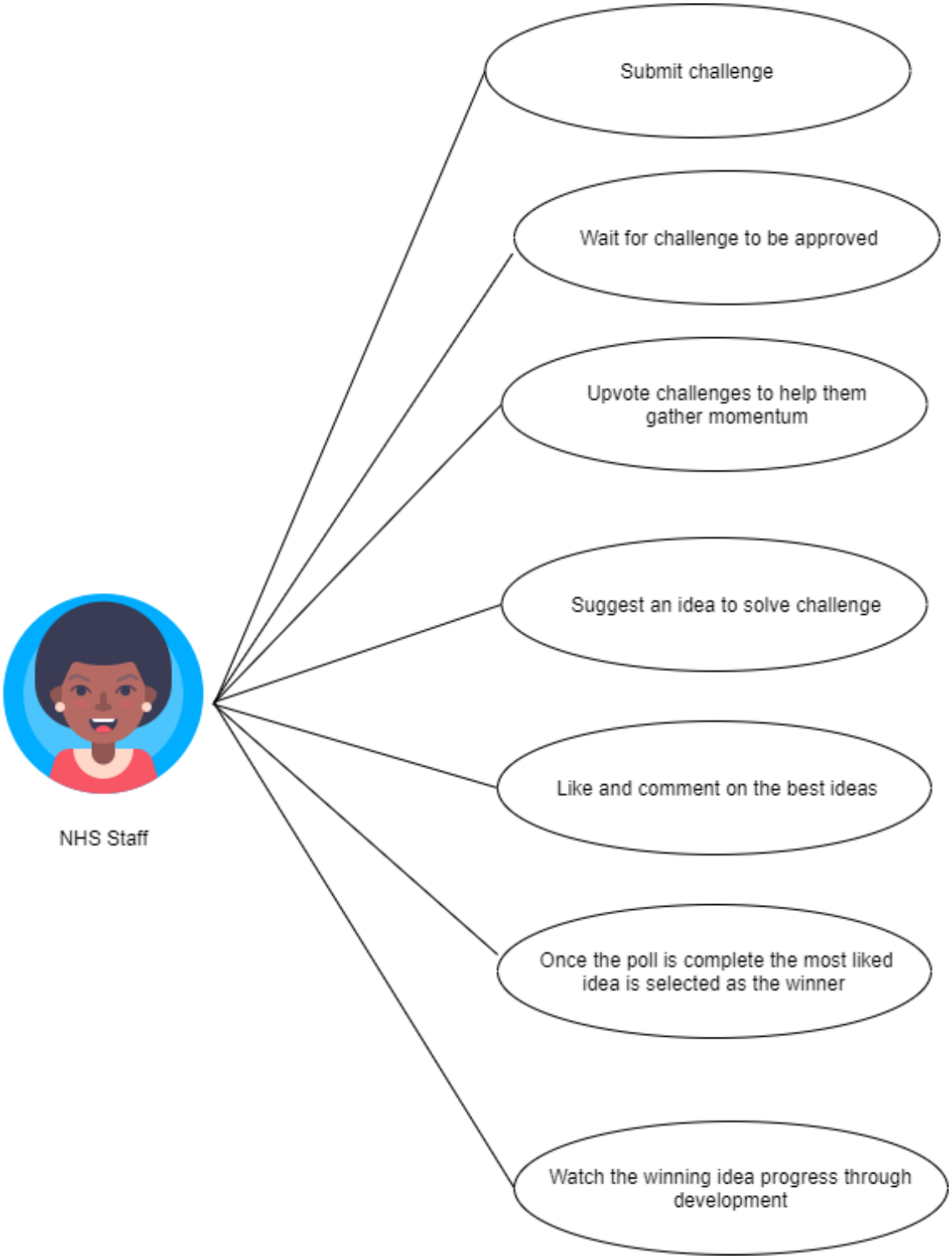

Figure 3.2: Standard NHS/Public User Use Case

Additional use cases can be seen in the appendix. Similarly, on the following page is a visual representation of this use case with a clear step-by-step summary of the journey from a user's perspective (Figure 3.2) [18].

Chapter 4

System Design

4.1 Outline

This section of the report documents the design of the NHS Idea Portal. It will begin by describing the architecture of the portal and then the critical areas of development before setting context for the following chatper on implementation.

4.2 High-level Architecture

The design and implementation of the system focused on two key objects - challenges and ideas. With this in mind, the architecture of the system placed them the center to make sure that they operated and exchanged well within the confinesof the system as a whole.

Implementing the back-end involved creating a PostgreSQL database and establishing a relationship within Django using models and migrations for updates to database tables [1]. The models allowed me to define criteria including the character limit on challenge or idea descriptions and the ability to upload accompanying images. Users would submit ideas and challenges that would be stored in the database and published in the management panel of the application (visible to Challenge Managers).

When submitting ideas, a GET request would be made to the ORCHA API (if the organisation has this setting active) and this would validate incoming ideas against existing technical solutions to prevent duplicates or similar applications being made. After ideas are approved they are given an 'open' stage status indicating that the idea has been selected for development and is in the elementary stages. Django models were also used to build the Django-REST API which published this information externally from the core application (Ideas Portal) [16].

Fugure 4.1 conveys how selected ideas would be published to the local API and were made visible in the Ideas Hub section of the application where users can track and idea as it progresses through the pre-determined stages of development. Once approved, these would update the status of the objects in the database and activate polls.

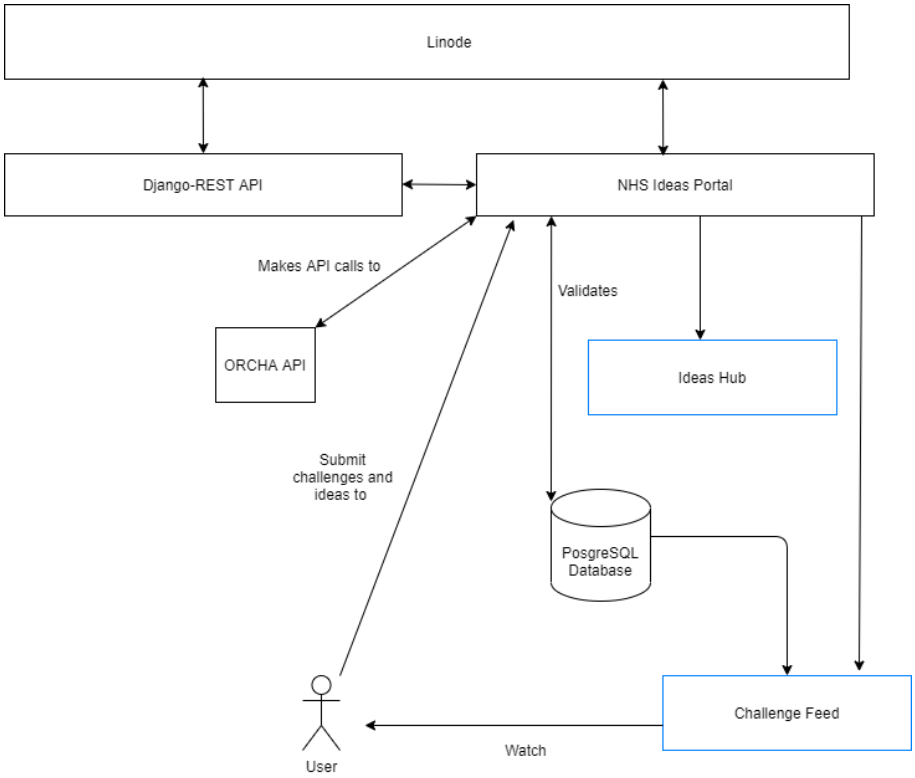

Figure 4.1: High-level view of ideation process

The blue boxes represent pages within the Idea portal where functionality is implemented heavily to interact with components and data at the highest level.

4.3 System data Flow

The system data flow diagram below (Figure 4.2) captures how data is transferred through the system through the journey and the role of each user-type throughout the process. This was created to identify the main components of the system whilst also capturing and conveying how data would flow and transfer between them.

This was a significant part of the design since it acted as the foundation of the projects a whole and was a frequent point of referral throughout implementation [11] [4]. Structurally, this provides a more detailed view of the ideation process than the architectural view in the previous section and effectively shows how ideas and challenges were handles and interacted with by different user types and sections of the database.

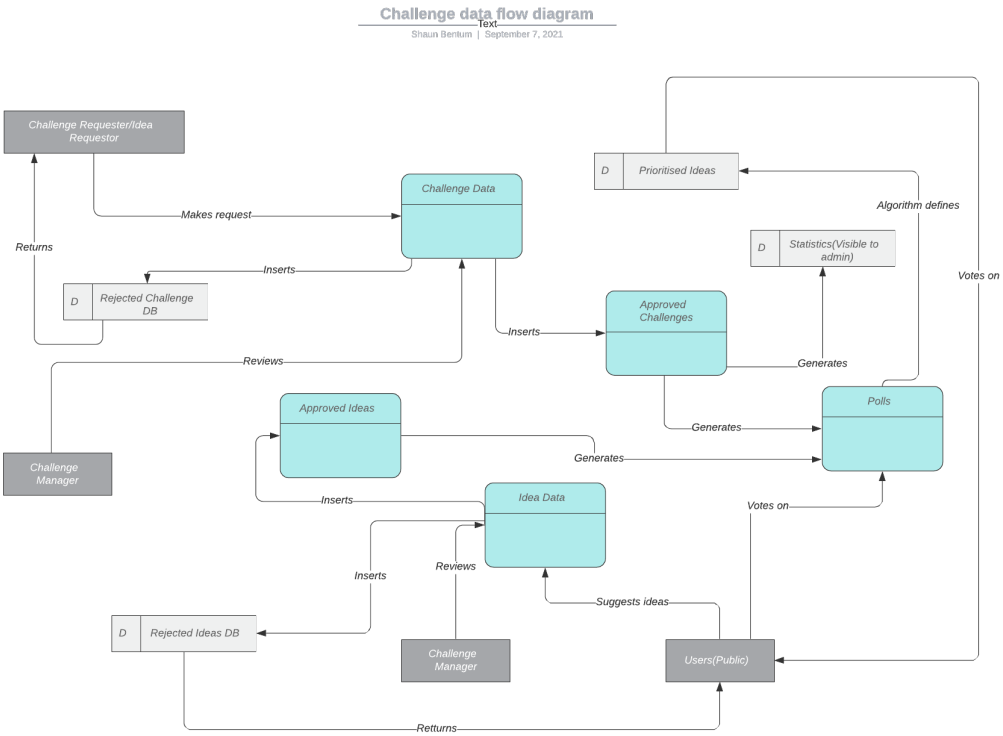

Figure 4.2: System Data Flow Diagram

4.4 Site maps

Figure 4.3 details the original site map design for the portal. All standard user types (NHS User, Public User) were modelled using this site map although changes were made during implementation in line with user feedback and also to suit NHS traditional styling and succinct multi-page forms.

After signing in each user is presented with a custom dashboard for the organisation they have signed into with a summary of org-wide statistics and their latest notifications. There is a navigation bar that allows users to easily navigate from here to the challenge feed where they will spend most of their time in the portal brainstorming challenges and ideas to these. They are also able to view more information on both object and like/dislike as well as making comments. All users have a profile page and from here they can edit their information and profile picture [9].

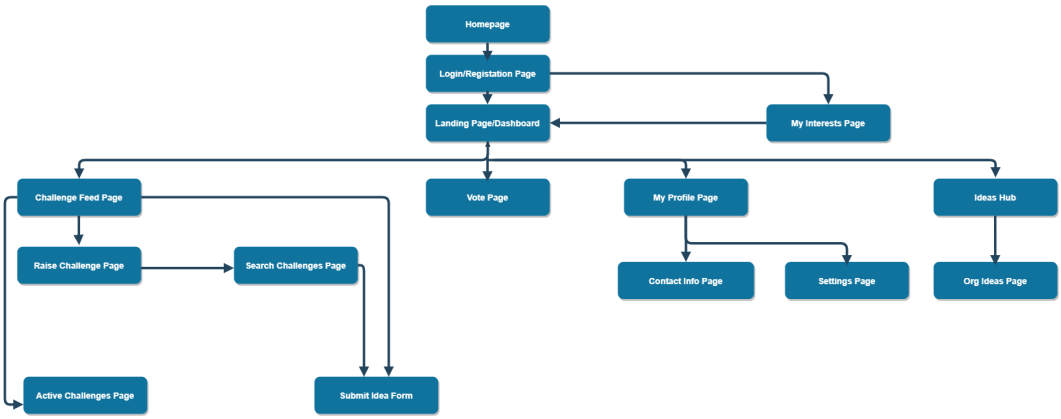

Figure 4.3: Standard User Site Map

Comparatively, the challenge managers and admins have a more options to correspond with their higher-level permissions (Figure 4.4). In addition to being access the pages and views that standard user-types have, they are also able to view the management panel where they can approve or reject ideas and challenges as well as being able to see and search for polls that have now expired.

Each tab has multiple levels of navigation and flexibility to multitask and view immediate computation of tasks in the system with live feedback [17]. This is

particularly useful to challenge managers given that they would have to manage entries in the portal but also partake in polls themselves.

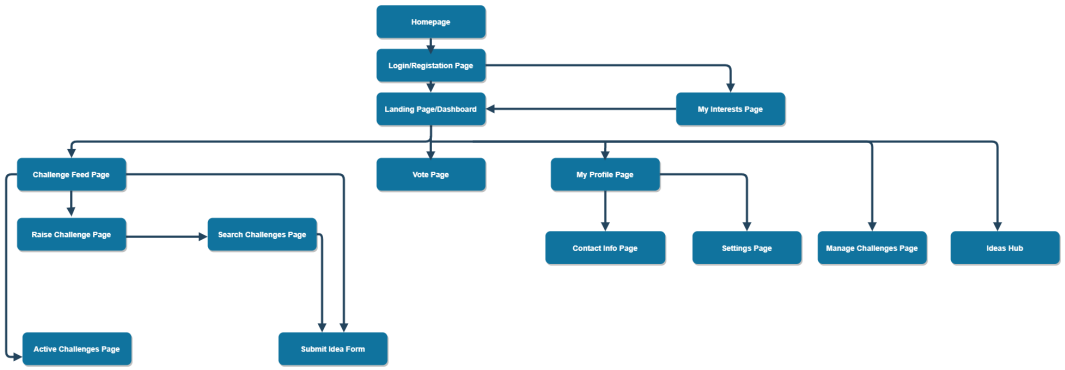

Figure 4.4: Advanced User Site Map

Chapter 5

Implementation

5.1 Models

For each of the required database objects I created a model and linked them together when necessary via foreign keys and many-to-many relationships. The model objects can be accessed for example, using the model 'Post' using Post.objects.all(). The models allowed for control over the PostgreSQL database from within django and I was able to establish a connection that would allow the objects from within the portal to be stored and updated in the db.

I made use of the commands 'python manage.py showmigrations' to view changes to database tables, 'python manage.py makemigrations' to confirm the changes and 'python manage.py migrate' to complete updates of the tables.

5.2 Organisation set-up

The requirement that the portal service a range of organisations within the NHS demanded that I factored contained spaces for ideation into my design - essentially creating portals with the overall portal. This was achieved by granting admins and portal managers the ability to create new organisations and customise them accordingly from the admin panel.

This feature made the implementation much more complex as it required me to make my code very dynamic and flexible to adapt to new organisations that may be added once the site was live and to also contextualise portal information for logged in users. The client also requested that portal managers and admins be granted the ability to switch the API and custom org-forms on/off when required - meaning that org-wide settings would be required.

```
        <h2 class="heading-section mb-5">Select a portal to sign in to</h2>
    </div>
    <div class="col-md-12">
        <div class="featured-carousel owl-carousel">
            {% for org in orgs %}
            <div class="item">
                <div class="row justify-content-center">
                    <div class="col-md-11">

                        <div class="testimony-wrap d-md-flex">


                            <div class="img" style="background-image: url(../../static/org_images/{{org.image}});"></div>


                            <div class="text text-center p-4 py-xl-5 px-xl-5 d-flex align-items-center">
                                <form method="post" action="">
                                    {% csrf_token %}
                                <div class="desc w-100">
                                    <p class="h3 mb-5">"{{ org.description}}"</p>
                                    <div class="pt-4">
                                                                                <p class="name mb-0">{{ org.name }}</p>

                                        {% if not request.user.is_authenticated %}
                                        <br>
                                        <a href="{% url 'blogfeed_main' org.slug %}">Enter anonymously</a>
                                        <br>
                                        <br>
                                        <a href="{% url 'auth_username' org.slug %}">Register an account</a>
                                        <br>
                                        <br>

                                        <a href="{% url 'logout' %}">Logout</a>
                                        <br>

                                        {% endif %}
                                        <br>
                                        {% if request.user.is_authenticated  %}
                                        <a href="{% url 'blogfeed_main' org.slug %}">Enter</a>
                                        <br>
                                        <br>
                                        <a href="{% url 'logout' %}">Logout</a>
                                        <br>
                                        {% endif %}


                                    </div>
                                </div>
                                </form>
                            </div>

                        </div>

                    </div>
                </div>
            </div>
            {% endfor %}
```

Figure 5.1: Org-hub HTML code

The figures above demonstrate from a code-level view how I used a for-loop to iterate through the organisations in the database and extract information into a carousel that would rotate and navigate to different portals granted the user had access to said portal. This was a key feature for creating unique user experiences depending on the orgs of affiliation.

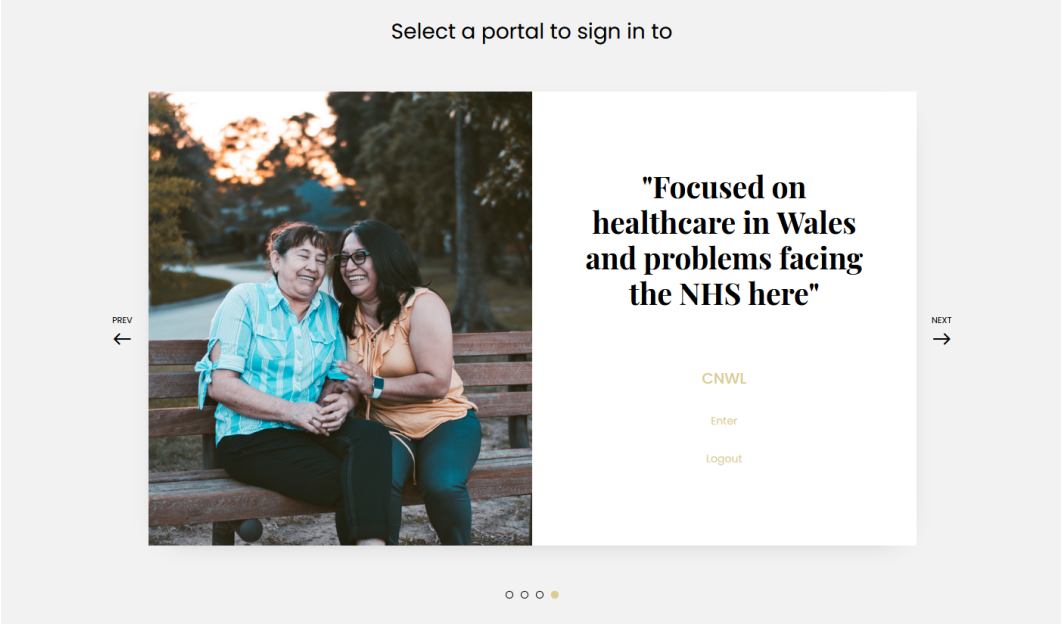

Figure 5.2: Org-hub UI view

5.3 Ideation

5.3.1 Challenges

Challenges were implemented to follow a standardized format that included 4 fields. Once a challenge is complete. A user must be authenticated in order to submit a challenge or they will be redirected to the access denied page. This measure is in place to prevent spam and reduce noise in the application, ultimately easing the responsibilities of challenge managers. A poll (Post) object is then created using the information inputted by the user into the challenge form.

For each submission users must specify a department that they believe best suits their challenge and this is critical for ensuring each challenge follows the correct pathway. Figure 5.3 details one of the main functionalities in the submit a challenge function. The function uses uses Django's built-in form functionality to create a challenge object from user inputted data.

```python
def submit_challenge(request, slug):
    if request.user.is_authenticated:
        form = ChallengeForm()
        logging.error(slug)
        org = slug
        orgobject = Organisation.objects.get(slug=slug)

        if request.method == "POST":
            form = ChallengeForm(request.POST)

            if form.is_valid():
                form.author = request.user
                challenge = form.save()
                challenge.author = request.user
                challenge.org_tag = orgobject
                cnwl = False
                challenge = form.save()
                Post.objects.create(author=challenge.author, title=challenge.title, severity=challenge.severity, department=challenge.department,
                challenge=challenge, description=challenge.description, org_tag = challenge.org_tag, image = challenge.image)

                return redirect('submit_challenge_successful', slug=slug)


        context = {'challengeform': form, 'org': org, 'custom_on' : orgobject.custom_form_on, 'orgslug' : slug}

        return render(request,'challenges/submit_challenge.html', context)
    else:
        return redirect('access_denied')
```

Figure 5.3: Submit challenge function

5.3.2 Polls

Once approved by a challenge manager and a start and end date is selected a poll is started and published publicly to all users affiliated with the challenge in which the org was published. Users are able to comment and upvote on poll to help gain traction and indicate their support of creating a solution for said challenge.

In the challenge feed users are then able to view polls ranked from the highest to lowest severity and categorised by the department specified in the accompanying 'Challenge' object.Figure 5.4 also demonstrates the dynamism of the code and ability to handle all forms and save attributes to the database tables as long as the mandatory fields are filled.

5.3.3 Ideas

Standard idea forms ask the user to fill a title and description as the only two mandatory fields. However, users also have the option to attach an image that could support the idea when being reviewed and help other users to interpret their solution. There are also notes and estimated cost fields which similarly exist to help support the credibility of the idea at hand.

For each poll users are able to view and search ideas by title or description. The code is dynamic here also to handle typos and capitalization. Ideas can be expanded to reveal more information and grant users the option of liking or comment the idea depicted in Figure 5.4. When a poll closes i.e. reaches its end date the idea with the most likes is automatically selected as the winner and the idea's stage field updates from None to 'open' - hence making it visible in the related Ideas Hub page for that organisation.

```
def like_view_idea(request, pk, slug, orgslug):
    idea = get_object_or_404(Idea, id=request.POST.get('idea_id'))

    liked = False
    if idea.likes.filter(id=request.user.id).exists():
        idea.likes.remove(request.user)
        liked = False
    else:
        idea.likes.add(request.user)
        liked = True
    return HttpResponseRedirect(reverse('idea_post', args=[orgslug, str(pk), slug]))
```

Figure 5.4: Like idea function

5.3.4 PRIDAR and organisation-specific forms

At an organisation level portal managers and admins are able to specify whether they would like to activate customised org forms. This enables two options - use the PRIDAR form adopted by CNWL or to customise their own forms with up to 17 user definable Boolean fields. Fields must be boolean i.e. True or False so data can be standardised also for ease of analysis when it comes to challenge managers reviewing forms rather than having to extensively dissect whether a forms information is viable or not based off the description, title or cost.

Once a poll is over and a PRIDAR idea is selected as winner it is published to the CNWL idea endpoint and at this point Trustnet accesses the API and begins the development of the idea. The view is simplified because the organisation-specific criteria is only require during challenge approval because there was a danger over over complicating the API by including all 17 user-definable fields.

Figure 5.7 demonstrates the update of the idea's stage from the API's endpoint and this change is reflected in the Idea Hub timeline thereby showing the successful implementation of the most important requirement asked of me by the client.

```
class IdeaSerializer(serializers.HyperlinkedModelSerializer):
    post = PostSerializer(read_only=True)
    org_tag = OrganisationSerializer(read_only=True, many=True)
    department = DepartmentSerializer(read_only=True)
    author = UserSerializer(read_only=True)

    class Meta:
        model = Idea
        fields = ('id', 'created_on', 'title', 'estimated_cost', 'description', 'stage', 'post', 'org_tag', 'department', 'author')
        read_only_fields = ('id', 'title', 'estimated_cost', 'description', 'post', 'author')
```

Figure 5.5: JSON Serializer for ideas in Django-REST framework

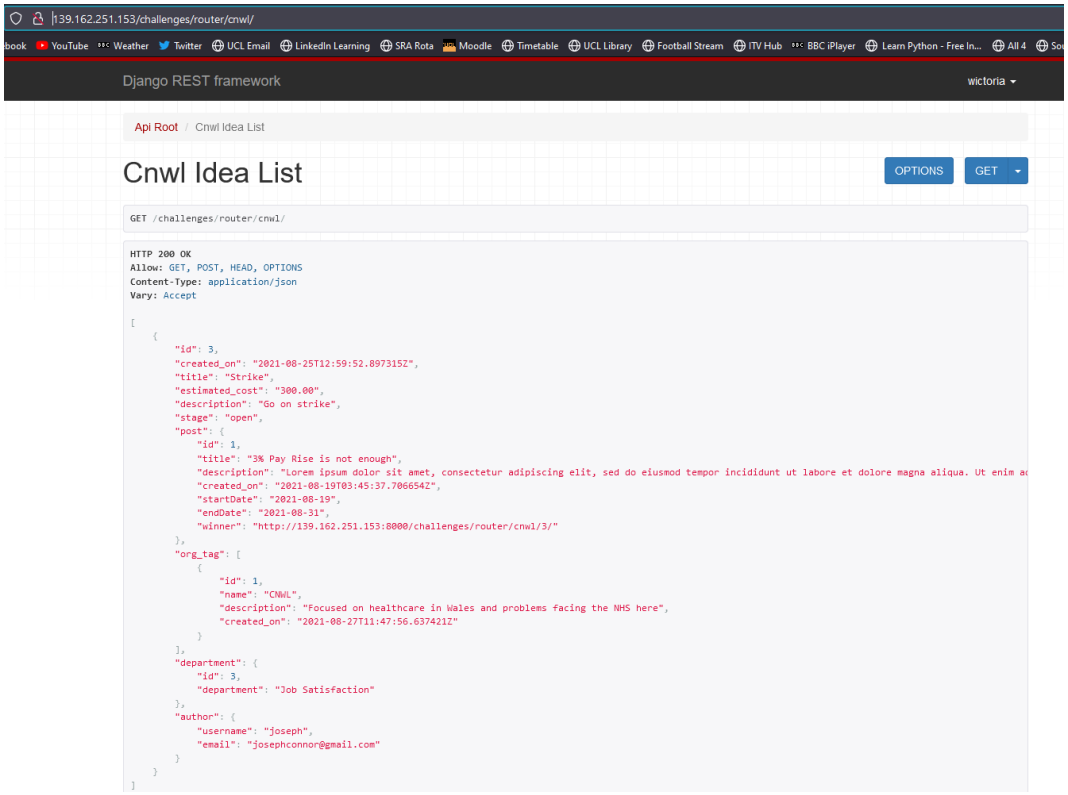

Figure 5.6: GET Request from API

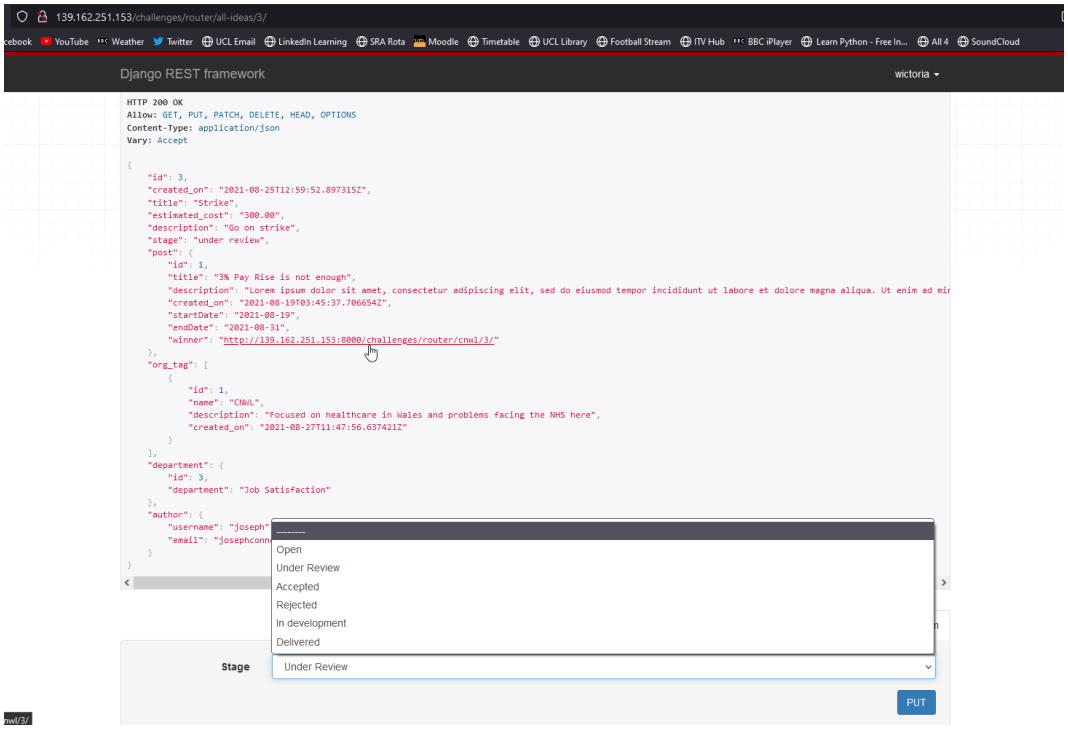

Figure 5.7: Succesful PATCH request to update idea stage to 'under review'

5.4 Back-end

5.4.1 ORCHA API

In earlier design iterations discussions were held about the use of ML algorithms, names the MLP to generate a word cloud to refine and compare new ideas agonist existing ones. However, this proved to be a complex and expensive process. As a result, a method was used to data traversal to extract the key words from incoming ideas and feed them back into the API to perform calls.

The ORCHA can be used to query ORCHA's review data using Capabilities, the audience the app was Designed For, Sub-Categories, Platforms, Costs and Countries of Origin. The capabilities perfectly suited the projects needs and if integrated with the portal would contribute to the quality of the final product [16], as well as being a significant requirement by the clients at NHS. The aim was to manipulate data from the API to refine ideas that would be produced in the platform.

5.4.2 Authentication

The API was simple - using credentials provided by a contact at ORCHA, I was able to authenticate and generate an access token which was the required in the header to make other requests.

```
if activate_api:
    # authentication credential for ORCHA API
    data = '{"username" : "cnwl", "password":"K2Q5!ZqnJ!#RYV"}'

    # header configuration to define format of data
    headers = {
    'Content-type':'application/json',
    'Accept':'application/json'
    }

    # authentication end point call to generate and access token to peform other REST API calls
    response = requests.post("https://app-library-builder-api.orchahealth.co.uk/api/orcha/v1/Token/Authenticate", data=data, headers=headers)
    # create a JSON from the string generated from the response
    body = json.loads(response.text)
    # parse the JSON to extract the access token and save as its own variable
    access_token = body['result']['accessToken']
```

Figure 5.8: Authentication of ORCHA API in Django

However the dependency on ORCHA and its approval by the NHS meant that there was a delay to receiving the said credentials and this to some degree limited the extent of data traversal. In addition to this, the documentation was limited and inaccurate meaning that I had to use problem-solving skills and existing knowledge of APIs to create functions that were capable of interacting with the external api using the requests library in Django.

5.4.3 Category Search and Keywords

The figure below displays code that was used to extract categorical data from the API that would later be used as search terms.This was critical to delivering the requirements of the application and understanding which areas ORCHA already had solutions for.

From the data returned by the call it is clear that ORCHA provides solutions tailored to specific illnesses such as cancer, ADHD etc. The follow-up function would seek to split these categories into individual words e.g. 'Back pain' into 'back' and 'pain' to increase the search capabilities and the likelihood of finding suitable matches.I made use of the built-in ORCHA search algorithm to search for similar applications using the key words as input in the GET request.

```python
# configure another header that uses the access token from the authentication end point
headers_2 = {"Authorization": "Bearer " + str(access_token)}
headers_3 = {
'Content-type':'application/json',
'Accept':'application/json',
"Authorization": "Bearer " + str(access_token)
}
# sub-category endpoint call to extract all categories that ORCHA use to define its applications
response_2 = requests.get("https://app-library-builder-api.orchahealth.co.uk/api/orcha/v1/SubCategory/GetSubCategories", headers=headers_2)
# create a JSON from the string generated from the response
body_2 = json.loads(response_2.text)
# parse the JSON to extract the list of categories
categories =  body_2['result']
category_list = []
# append the list of categories to a list format to make it easy to operate on
for category in categories:
    category_list.append(category['subCategoryName'])
```

Figure 5.9: Access token used in GET request to retrieve application categories in ORCHA

If the user inputs a term into the title or description that matched a key word then a search would be performed to extract all applications from ORCHA that were returned in the search algorithm.

5.4.4 Idea Validation

A more refined list is defined and key-words are appended to it for a more refined search based on data inputted by the users. For each key-word a search is performed and the resulting applications are appended to a list which is traversed through to extract information. The next figure (5.10) captures the method for idea validation, the outcome is then logged for debugging if required.
The input from the user is translated into a format that can be compared against the key-words extracted from the category in the previous section. Set methods are used to check for and compare between words.

For each key-word a search is then performed within the ORCHA framework. The search uses ORCHA's own search algorithm to return applications based of these words. A list is then indexed that extracts the app's name, publisher, description, download link and platform so that these can be passed to the user's view in the front-end. An example can be seen below, based off an idea where the user entered the word 'cancer' - which was recognised as a keyword by ORCHA.

```python
# get user idea title
compare = current_idea.title
# get user idea title
compare_desc = current_idea.description
title_list = []
description_list = []

# append key words form title to list
for word in compare.split():
    title_list.append(word)
logging.info(title_list)

for word in compare_desc.split():
    description_list.append(word)

# initiate state variable
is_similar = False
category_names = []

category_list_lower = [item.lower() for item in category_list]
title_list_lower = [item.lower() for item in title_list]
description_list_lower = [item.lower() for item in description_list]

# if the keywords from the idea match any sub category areas set state to True and create a list of the similar terms
existing_ideas = ""

if (set(category_list_lower) & set(title_list_lower) ) & set(description_list_lower):
    logging.info("This idea could be similar to an exisiting solution")
    is_similar = True
    logging.info("MATCH ALERT")
    category_names = list(set(category_list_lower) & set(title_list_lower) & set(description_list_lower))
    logging.info(category_names)
else:
    logging.info("This idea is not similar to an exisitng solution")

# append key words from title and description to list
keywords = []
keyterms = []
for name in category_names:
    name.split()
    keywords.append(name)
logging.info(keywords)
```

Figure 5.10: Keyword generation and comparison against words users input into the form

```python
# append key words from title and description to list
keywords = []
keyterms = []
for name in category_names:
    name.split()
    keywords.append(name)
logging.info(keywords)

# for each key word identified search in ORCHA for an application with the same keyword in title

for category in keywords:
    data_2 = '{"searchTerm": "' + category + '","pageNumber": "1", "pageSize": "12", "costIds": [],"capabilityIds": [],"designedForIds": [],"countryIds": []}'
    response_3 = requests.post("https://app-library-builder-api.orchahealth.co.uk/api/orcha/v1/Review/SearchPagedReviews", data=data_2, headers=headers_3)
    # create a JSON from the string generated from the response
    json_str = json.loads(response_3.text)
    # parse the JSON to extract the applications
    existing_ideas = json_str['result']['items']
    counter = 0
    # count the number of applications that the algorithm recorgnises as similar solutions
    for id in existing_ideas:
        counter = counter + 1

    existing_idea_names = []
    existing_publisher_names = []
    existing_descriptions = []
    existing_links = []
    exisiting_platforms = []

    # for each application extract the name, download link, platform etc.
    for idea in existing_ideas:
        existing_idea_names.append(idea['appName'])
        existing_publisher_names.append(idea['publisherName'])
        existing_descriptions.append(idea['description'])
        existing_links.append(idea['downloadLink'])
        exisiting_platforms.append(idea['platform'])

    zipped_values = zip(existing_idea_names, existing_publisher_names, existing_descriptions, existing_links,exisiting_platforms)
    logging.info(zipped_values)
```

Figure 5.11: GET request to extract relevant applications that could be similar or duplicates

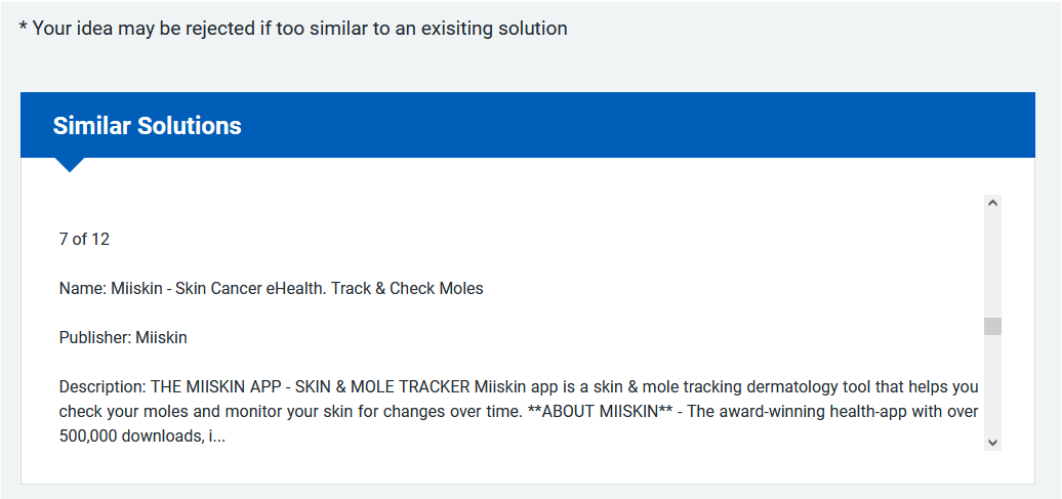

Figure 5.12: API integration to idea forms in front-end

Users can navigate directly from the portal to review the application and make a judgement as to whether it is a duplicate or similar to the idea that is currently being proposed. If so, they can terminate the idea submission and download an existing solution of their choice or alternatively complete the remainder of the form at the risk that it may be rejected if the challenge manager rejects the idea being proposed. This evidences the importance of tech in revolutionising the healthcare industry, especially APIs [15].

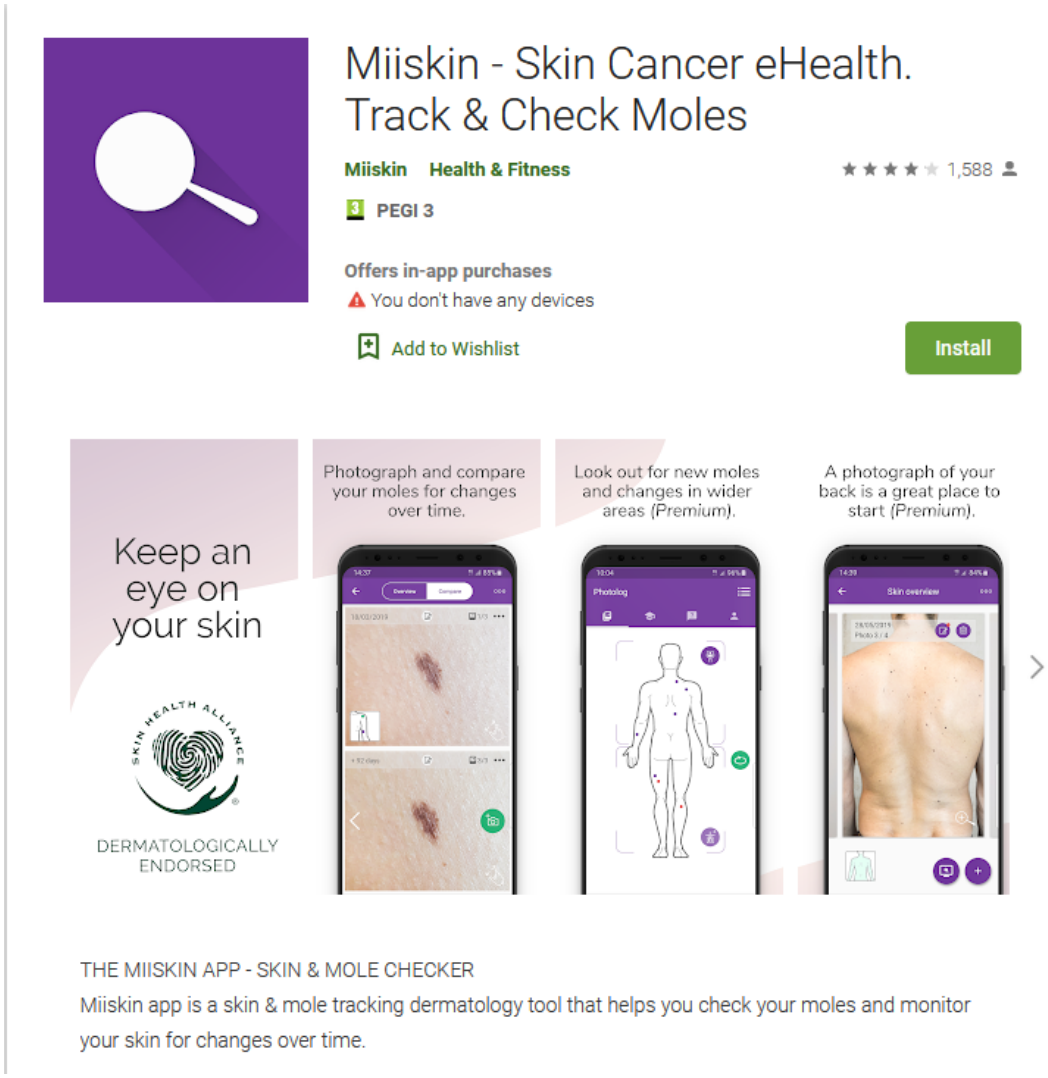

Figure 5.13: Link to app stores with more information and downloads

5.4.5 Email Verification

For security purposes it was important to verify users so that the ideation was limited to genuine users that sought to collaborate effectively in the portal with minimum noise. Users are sent a one-off activation link in email once they complete the registration that must be activated before they login. By clicking the link this updates the 'email-verified' flag on each user to True and grants them access to the portal.

```python
def activate_user(request, uidb64, token):

    try:
        uid = force_text(urlsafe_base64_decode(uidb64))
        logging.info("Your uid is: ", uid)

        user = Account.objects.get(pk=uid)
        logging.info(user)

    except Exception as e:
        user = None

    if user and generate_token.check_token(user, token):
        user.is_email_verified = True
        logging.info(user.is_email_verified)
        user.save()
        logging.info("good path")

        return redirect(reverse('activation_success'))

    return render(request, 'emails/activate-failed.html', {"user": user})
```

Figure 5.14: Link to app stores with more information and downloads

5.4.6 Deployment

The application is deployed to Linode. Please see Appendix B for the deployment manual. As mentioned in the project background, Linode was selected by the client due to affordability and seamless hosting with existing NHS/IXN platforms. I was able to ssh to the server as a root user and customise nginx and gunicorn files to be able to serve the portal once in production. Filezilla was also used in support to achieve CI/CD.

Chapter 6

Testing

6.1 Functional Testing

The purpose of functional testing is to ensure that the features deliver on their intended purpose and act as expected in context of the portal as a whole [10] [6].

6.1.1 Manual Tests

I performed testing manually on less frequent user journeys such as being able to comment on posts and the profile. The intention behind this was to achieve a quick summary on the status of more complex areas of the system that would be timely to automated using selenium. Furthermore, I was able to extensively test out the more fragile features that would require several lines of automated code and more complex/time-consuming scripts.Based on use cases and requirements, the following test cases were created:

6.1.2 Automated Selenium Tests

In support of the weekly sprint cycles I had established - at the close of every week I would dedicate some time to execute a QA test cycle [3][22]. During this period I would execute my regression testing to monitor if and how the week's develop-

ments had raised any bugs. I allocated different levels of severity for bugs with 'blockers' being those that impeded features and required fixes as soon as possible. In doing so, I was able to rigorously test the product throughout development and ensure that the key features and end-to-end user scenarios worked as expected.

I would also categorise any failing tests as either 'test failures' or 'bugs' with respect to errors in the automation code and the development code respectively. Advantages of this approach include - minimising the likelihood of bugs making it to production, quick feedback times, reduced manual effort/strain and ultimately higher test coverage. There is a risk associated with testing one's own application - the automated tests reduced this risk of human error by essentially acting an unbiased tester.

6.1.3 Regression Testing

After deployment to the Nginx server I conducted the executed the automated test cases to assess how well the portal functioned in production. Figure 6.1 is the script for the first automated test case. The test is intuitive and adapts to the exception handling in place surrounding user management by generating a random username and email since the system does not allow duplicates.

The test clicks through the user registration journey and returns a result in 8 second as opposed to 60 seconds+ if I had completed the test manually; highlighting its value during regression testing. I would execute all functional and unit test cases after major changes to the application to ensure that my pre-exisitng code still worked as expected. I would then assign a pass rate to each test case. The pass rate indicates how many times the test passed out of 10 executions. Tests with lower pass rates are indicative of bugs or poor automated code and this is represented in following table.

```python
def test_registration(self):

    # Create account
    self.browser.get('http://localhost:8000')
    createAccountBut = self.browser.find_element_by_xpath("//a[contains(text(), 'Click to create an account')]")
    createAccountBut.click()
    time.sleep(1.5)
    registerBut = self.browser.find_element_by_xpath("//span[contains(text(), 'Register with us')]")
    registerBut.click()
    time.sleep(1.5)
    usernameBox = self.browser.find_element_by_xpath("//input[@id='id_username']")
    gen_username = str(random.randrange(0,5000))
    usernameBox.send_keys('Random_user' + gen_username)
    time.sleep(1.5)
    emailBox = self.browser.find_element_by_xpath("//input[@id='id_email']")
    emailBox.send_keys('random' + str(random.randrange(0,5000)) + '@nhs.net')
    password1 = self.browser.find_element_by_xpath("//input[@id='id_password1']")
    password1.send_keys("Persona_123")
    password2 = self.browser.find_element_by_xpath("//input[@id='id_password2']")
    password2.send_keys("Persona_123")
    time.sleep(1.5)
    continueBut = self.browser.find_element_by_xpath("//button[@class='nhsuk-button']")
    continueBut.click()


    time.sleep(1.5)
    continueBut2 = self.browser.find_element_by_xpath("//button[contains(text(), 'Continue')]")
    chosenAge = self.browser.find_element_by_xpath("//input[@value='21-30']")
    chosenAge.click()
    continueBut2.click()

    time.sleep(1.5)
    continueBut3 = self.browser.find_element_by_xpath("//button[@id='submit-button']")
    chosenJob = self.browser.find_element_by_xpath("//input[@value='Yes']")
    chosenJob.click()
    continueBut3.click()

    time.sleep(1.5)
    chosenInterest1 = self.browser.find_element_by_xpath("//input[@value='culture']")
    chosenInterest2 = self.browser.find_element_by_xpath("//input[@value='leadership']")
    continueBut4 = self.browser.find_element_by_xpath("//button[@class='nhsuk-button']")

    chosenInterest1.click()
    chosenInterest2.click()
    continueBut4.click()

    time.sleep(1.5)
    logBackInBut = self.browser.find_element_by_xpath("//button[contains(text(), 'Login')]")
    logBackInBut.click()

    time.sleep(5)

    portalUsername = self.browser.find_element_by_xpath("//input[@name='username']")
    portalUsername.send_keys('Random_user' + gen_username)

    time.sleep(1.5)
    portalPassword = self.browser.find_element_by_xpath("//input[@name='password']")
    portalPassword.send_keys('Persona_123')

    portalLogin = self.browser.find_element_by_xpath("//input[@name='Login']")
    portalLogin.click()
    time.sleep(10)
    if self.browser.find_element_by_xpath("//header[@class='nhsuk-header']"):
        logging.error('ELEMENT LOCATED - Login Successful')
    else:
        logging.error('ERROR - element not located')
```

Figure 6.1: Automated Functional test case example (user registration)

6.1.4 Unit Testing

With TDD at the helm of my project, I would make sure unit tests were passing prior to picking up the next development task. Figure 6.2 captures some exemplar unit tests that I use to make sure my application was using the correct templates and redirecting to the correct pages. In doing do, I minimised margin for error and the the risk of bugs making it to production.

```
class UnitTestCase(TestCase):

    def test_home_homepage_template(self):
        response = self.client.get('/')
        self.assertTemplateUsed(response, 'authentication/home.html')

    def test_registration_template(self):
        response = self.client.get('/register/')
        self.assertTemplateUsed(response, 'userauth/register.html')

    def test_sign_in_template(self):
        response = self.client.get('/sign_in/')
        self.assertTemplateUsed(response, 'userauth/sign_in.html')

    def test_hash_form(self):
        form = CustomUserCreationForm(data={'text':'hello'})
        self.assertTrue(form.is_valid())
```

Figure 6.2: Unit test cases

6.1.5 System Integration Testing

After deployment to the Nginx server, I conducted the executed the automated test cases to assess how well the portal functioned in production. This constituted of executing my automated scripts in production and viewing results before debugging. This method proved quick and effective for recognising bugs and highlights again the importance of testing to the quality of the final application I adopted GHERKIN and CUCUMBER syntax when writing my test cases for ease of understanding and a clearly defined goal for each of the test cases [10].

Test Cases		
Conditions	Pass Rate	Bug or Test Failure?
1. GIVEN I am a public user WHEN I try to complete registration for the portal THEN I am able to register successfully and login with the specified credentials	100	N/A
2. GIVEN I am logged into the portal as any user type WHEN I submit a challenge THEN the challenge is submitted to a challenge manager for review	100	N/A
3. GIVEN I am logged into the portal as a challenge manager WHEN I approve a challenge AND I set a start and end date THEN a poll is created in the appropriate portal	80	N/A

Conditions	Pass Rate	Bug or Test Failure?
4. GIVEN I am logged into the portal as a challenge manager WHEN I approve a challenge THEN a poll is created in the appropriate portal	80	N/A
5. GIVEN I am logged into the portal as any user type WHEN I suggest an idea for the challenge THEN the idea is submitted to a challenge manager for review	40	Test Failure
6. GIVEN I am logged into the portal as any user type WHEN the poll reaches its end date THEN the idea with the most likes is selected as the winner	60	Bug

6.2 User Acceptance Testing

I allocated a week to roll out the application and perform user acceptance testing (UAT). This provided an opportunity for real users to evaluate how well the initial goals and requirements had been met, point out missing functionality, and identify any critical bugs that had been missed in earlier testing cycles. I then factored in these responses - keeping track of what was well received and areas for improvement before outlining my tasks for the following week. Using qualitative data analysis based on a Likert scale I was able to attach weighting the opinions of my test group and prioritise my tasks based on these [13].

FUNCTIONALITY					
	Yes	No	If No, Specify...		
1. The website loads fast					
2. The links take you to the relevant pages					
3. The buttons are looking and working fine					
4. The functions accept the correct inputs and give the correct output *Test each and every field on the website to make sure only allowed values for the data field are accepted.					
5. The filters return the expected results					
6. Search results are correct					
7. Uploaded documents/images open properly					
DESIGN					
	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
1. I find the website design to be attractive					
2. The website has a clean and simple presentation					
3. The images on the website are aesthetically pleasing and relevant					
4. The interface (colours, layout, font-styles, buttons) is consistent across all pages					
5. I am able to differentiate links easily					
CONTENT					
	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
1. The content is clear and precise					
2. The content is easy to understand					
3. The content is correct without any spelling or grammatical errors					
4. The content is meaningful and relevant to the pages					
USABILITY					
	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
1. The website is easy to use					
2. It is easy to navigate within this website					
3. The menu items clearly tell me where they will take me					
4. I felt the website responded in a consistent and predictable way					
5. I can use the website without written instructions					
6. I think most people would learn to use the website quickly					
7. I can recover from mistakes quickly and easily					
8. I was able to easily find all the information I was looking for					
9. The website contains easy filtering options					
GENERAL					
1. What was your overall impression of the website?					
2. What did you like best about the site?					
3. What did you like least about the site?					
4. What, if anything, caused you frustration?					
5. If you could change one thing about the website, what would it be? Why?					

Figure 6.3: UAT Survey

An underlying common theme from the testing was that users felt the portal should have more aesthetic UI elements. The suggestions were made in pictures and animations, however, my challenge was trying to achieve this whilst still conforming to the highly accessible and easy-to-use NHS styling.

"It looks clean. I found it slightly confusing in terms of what I might find under the different headings of challenge feed, submit a challenge or challenge history. Maybe a one liner somewhere to explain what the site is about and why you would visit might help?" - Un-named Tester (CNWL)

I would take user feedback into account when bug-fixing and trying to implement any features that were highly suggested where possible. This demonstrates again the user-centred focus of my projects and as a result it has delivered on the expectations of the users [17].

"I could see on each challenge a button called 'suggest idea', but I couldn't work out if this was the same as 'submit idea' or a related to the challenge itself. I found it confusing"- Un-named Tester

6.2.1 Results

In accordance with TDD, I had to ensure that my test cases had passed before developing the next feature [6]. Hence, from the first sprint cycle the test cases detailed in the earlier section had passed on repeated execution. Nevertheless, some persistent bugs arose in other areas, especially post-deployment surrounding the serving of user images. From the first to the last spring cycle there was a significant increase in successful execution of test cases and this can be attributed to a period of bug-fixing and the refinement of selenium automation scripts.

Ultimately, the client was very satisfied with the completed ideation portal, placing value of the user-focused design and my interpretation of consistent feedback and changes requested. As well as delivering the key functionalities to a top standard they praised my ambition and nowse to modernise the application and include competitive features to improve the UX elements.

Chapter 7

Conclusion and Project Evaluation

7.1 Achievements

In the introductory chapters of the report, the project was contextualised and the aims were outlined clearly. In this section I will critically evaluate the project from design to delivery and assess whether it achieved the key goals, The key goals can be succinctly summarised as:

- The portal shall include a way to compare user feedback against existing responses and ideas (API) to validate that it is a new idea
- The portal shall enable all users to submit and respond to challenges that have been posted with their own ideas
- The portal shall assess PRIDAR criteria for some ideas

After discussions with the clients at CNWL it was made clear that these 3 functionalities were central to the overall success and usability of the portal. Based on the outcomes of rigorous test cycles the inference can be made that the portal has met these requirements to a high standard. Furthermore, the application is fully deployed and accessible on the internet via the URL provided in the User Manual section of the appendix.

7.2 Fulfillment of Personal Goals

Whilst the importance of the project meeting its goals should not be downplayed, it is important to reflect on my own experiences as a software engineer. Initially, I sought to challenge myself by setting the task of building an application from start to finish with an unfamiliar framework. The successful delivery of the NHS Ideas Portal has meant that I have indeed achieved this whilst also abiding to the principles of TDD; minimising defects to produce as high a quality product that I could. I also wanted to expand my own knowledge and learn about principles of web development including simplicity of design and the importance of navigation that I had not come across before. Valuable experience was gained on the business-side too; speaking frequently with CNWL, interpreting the organisation's requirements and demonstrating progress weekly - key skills in software engineering [3] [19].

In completing the project, I was able to understand the importance of user-focused design and structured planning. Django and Django-REST's ease of use and powerful capabilities were the foundation of development and I now feel very comfortable using both frameworks. I also learnt about the effectiveness of agile methodologies in modern-day web development; by iterating sprint cycles I was able to constantly review my progress and minimise the margin for error.

Finally, the successful integration of the ORCHA API and build of the NHS Idea Portal API significantly expanded my knowledge in the area which was also one of my early goals. What was a daunting task to begin with, I was able to complete through strategy, extensive research and hard work. Crucially, I value the importance of problem-solving now even more; a concept central to computational sciences that I look forward to exploring further in my career.

7.3 Critical Evaluation

Although the application delivered it's expected functionality, it is also important to recognise and highlight areas of improvement. A delay in gaining access to authorisation credentials for ORCHA limited the amount of functionality that I was able to integrate in the remaining time. This was the most significant challenge because as well as being granted access late into July, the API documentation wasn't correct so I had to explore myself how to format requests so that I could extract and compare them against user-inputted data. I overcame this hurdle through critical thinking and debugging; ultimately delivering on the requirement.

In hindsight, the NHS Ideas Portal API could have also offered more functionality such as filtering and ordering to avoid feeling too restrictive to users. There is also a risk that it becomes more difficult to operate as more ideas populate the API.

On review, deployment is a phase that I would have planned from the beginning as there were a number of unexpected and time-consuming bugs that flagged up after I moved the site to run on the Nginx server. I would have also liked to use Docker to contain sections of that app and simplify the re-deployment process for future developers.

Critical evaluation also involves crediting the good decisions I made throughout the project [22]. For example, through frequent regression testing and a period of bug-fixing I believe this demonstrated my thorough understanding of the importance of testing applications before reaching production to minimise bugs [10]. This principle is at the heart of the TDD approach and I believe contributes significantly to the quality of the final product. The decision to automate tests from the early stages of development meant that I was quick to identify and resolve issues throughout and will be useful when adding new features to the site in the future.

Overall, there are definite areas for improvement which are explored in more detail

in the next section but as a whole the project has delivered on its requirements and has managed and pushed the boundaries of my knowledge and practical skills.

7.4 Future Work

Given more time, I would have included more of my 'should have' and 'could have' features from the Moscow requirements and expanded my regression suite to cover areas of functionality including REST API calls. Time constraints meant that I had to test these manually which was quite a strenuous and repetitive process [17]. Automation of these tasks would have eased the workload here and increased capacity for testing and implementing extra features. In the future it would be great to use the interests of users to influence the type of content they view in their feeds and allow admins and portal managers to view a wider range of statistics and filter these by role or age group for example.

The future work for the project can be separated into two categories. Firstly, there is the unfinished work from the project due to time constraints. This would include:

- View profiles and allow comparison directly between users
- View more complex statistics and offer graphical views as well as current numerical ones
- The portal shall allow users to login by scanning a QR code
- The portal shall include more polling in other areas (User feedback and rating of the site)
- The portal shall notify users via push notifications, e.g likes and comments on their challenges (move to functional)
- The portal shall notify users via push notifications, e.g likes and comments on their challenges (move to functional)

And secondly, those features that were always beyond the scope of the project:

- The portal shall provide accessibility features for impaired users
- The portal shall allow users to login by scanning a QR code

7.5 Conclusion

In conclusion, the NHS Ideas Portal was a success; delivering on its promise of a space for organisations to brainstorm and collaborate effectively to deliver solutions. From an academic standpoint, the learning I have undertaken throughout the project is invaluable and I would like to extend my gratitude to UCL, IXN and the NHS for granting me the opportunity to engage with and take on such an interesting and exciting project.

As a software engineer, I have learnt and become skilled in the use of Django and Django-REST frameworks, successfully deployed a live application and executed a project plan that I myself designed; all new experiences for me. I was fortunate enough to gain hands-on experience working with APIs and became astute in web development, which I may not have had the opportunity to do so otherwise.

The experience has most importantly shown me the value of agile methodologies and TDD in producing high-quality applications and it is a principle that will definitely be incorporated into my future work. I am proud of the project outcomes and I trust that the resulting application will serve the NHS and its users well.

Bibliography

[1] Jacob Kaplan-Moss Adrian Holovaty. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2007.

[2] Lisa Bodell. Why t-shaped teams are the future of work. 2020.

[3] Alex Campbell. Agile: Essentials of team and project management. manifesto for agile software development. 2020.

[4] John Carroll. Five reasons for scenario-based design. *IEEE Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

[5] Justin Ellingwood. How to set up django with postgres, nginx, and gunicorn on ubuntu 18.04, 2018.

[6] K. Beck Three Rivers Institute. Test-driven development by example, 2021.

[7] Mark Lavin. *Lightweight Django: Using REST, WebSockets, and Backbone*. O'Reilly Media, Inc, 2014.

[8] Travis Lowdermilk. *User-Centered Design: A Developer's Guide to Building User-Friendly Applications*. O'Reilly Media, 2014.

[9] Jakob Nielsen. 10 usability heuristics for user interface design. 2020.

[10] E. Dornenburg J. Nilsson. Test driven development (tdd) – an overview. 2016.

[11] Donald Norman. The 6 principles of design, a la donald norman. *Enginess*, 2014.

[12] The Medic Portal. Challenges facing the nhs and current nhs issues, 2021.

[13] Dan Radigan. Project management with agile principles. 2021.

[14] M. Rouse. Test-driven development (tdd), 2017.

[15] Chanaka Fernando Sachini Samson. Transforming the healthcare industry through api marketplaces. 2019.

[16] Raivat Shah. Introduction to rest apis. 2020.

[17] Preece Jennifer Rogers Yvonne Newark: Wiley: 5th ed: 2019 Sharp, Helen. *Interaction design: beyond human-computer interaction*. Wiley, 2019.

[18] Teo Siang. Personas – a simple introduction. *Interaction Design Foundation*, 2021.

[19] Ian Sommerville. Software engineering 9th edition. *ISBN-10*, 137035152:18, 2011.

[20] Techopedia.com. Functional testing, 2021.

[21] William S. Vincent. *Django for APIs: Build Web APIs with Python and Django*. WelcomeToCode, 2018.

[22] VisualParadigm. What are the three amigos development strategy in agile?, 2021.

[23] Tsvi Vinig Yu Mu, Bart Bossink. Employee involvement in ideation and healthcare service innovation quality. 2017.

Appendix A

User Manual

This section details all the key pages of the application with screenshots and instructions to highlight the main functionalities of the app.

A.1 Login Details

Pre-inserted data on PostgreSQL. This can be used to login into the portal when required. Please bear in mind that users are only able to log into portals that they are affiliated with.

Site URL - http://139.162.251.153/

Table A.1: Login details

Account Type	Username	Password	Affiliation
Admin	admin	Persona123	all
Challenge Manager	challenge_manager	Persona123	Public, CNWL
Portal Manager	portal_manager	Persona123	Public, CNWL
NHS Staff	nhs_user	Persona123	all
Public User	public_user	Persona123	NHS-London NHS South-London

- Advanced users (Admins, Portal Managers and Challenge Managers) can complete management tasks roles at:

  https://139.162.251.153/ideaportal-management

- API GET request can be made by authenticating with any of the above login details at the endpoint:

  139.162.251.153/challenges/router/all-ideas

  ...to view all winning ideas

- API PATCH, PUT, DELETE requests etc. can be made by authenticating with any of the above login details at the endpoint:

  139.162.251.153/challenges/router/all-ideas/id?

- CNWL-specific ideas accessed by authenticating with any of the above login details at the endpoint:

  139.162.251.153/challenges/router/cnwl

  ...to view all winning ideas under the CNWL organisation banner

A.2 Sign-in and Registration Pages

Figure A.1: Sign in page

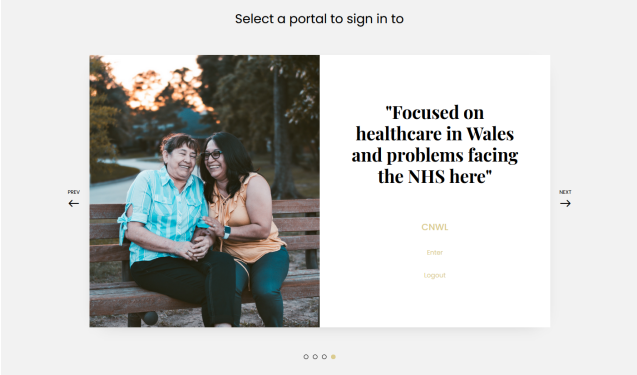

Figure A.2: Enter Portal Page

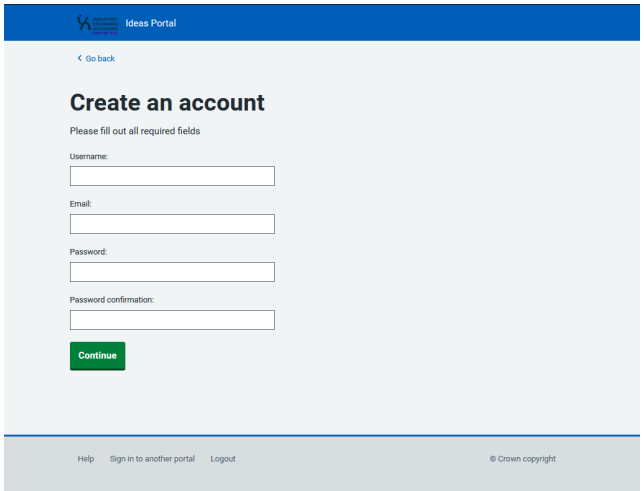

Figure A.3: Registration 1 - Username and password

Standard NHS multi-step format for accessibility and consistency with existing platforms. Users must fill all fields to proceed.

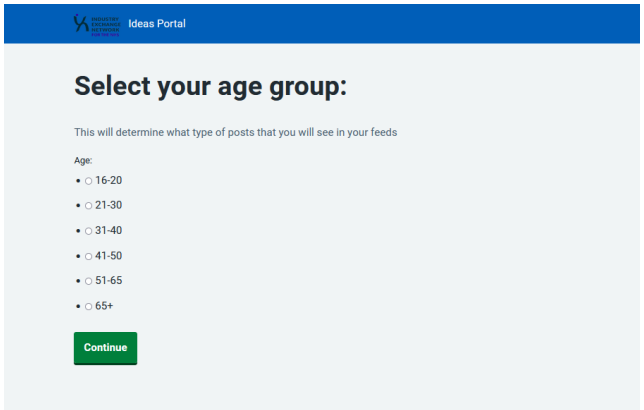

Figure A.4: Registration 2 - Age group

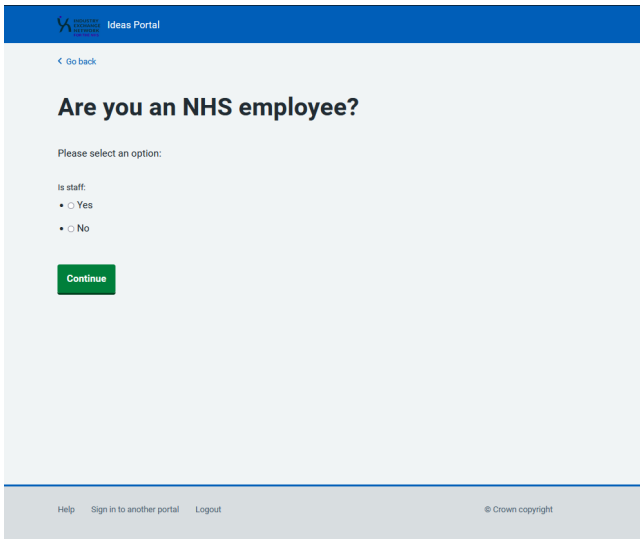

Figure A.5: Registration 3 - NHS staff member?

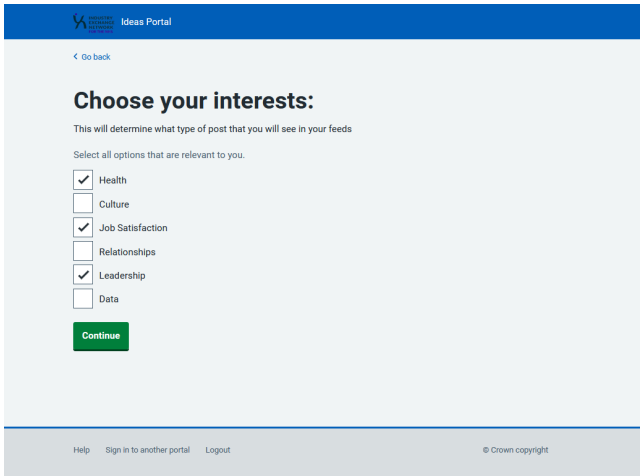

Figure A.6: Registration 4 - Select your interests

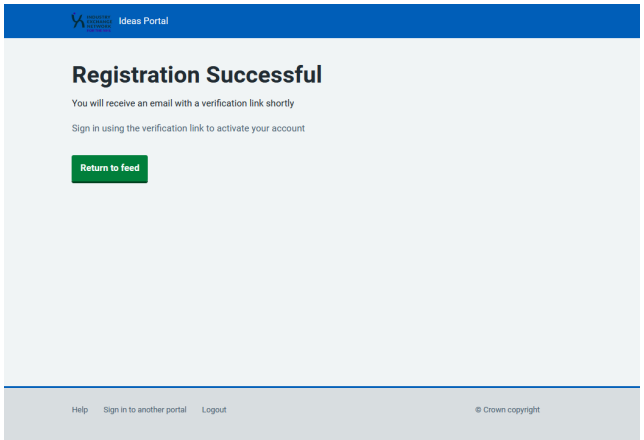

Figure A.7: Registration Successful

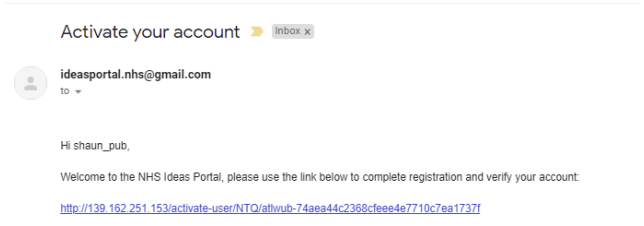

Figure A.8: Verification email

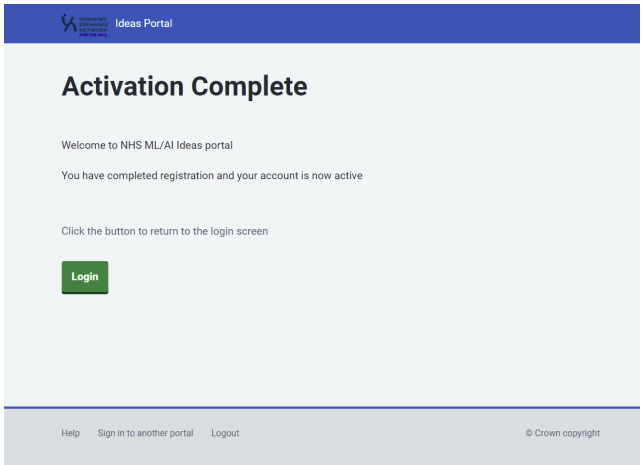

Figure A.9: Activation complete

New users receive an activation email with the credentials that they registered with. Users must click the activation link in order to access the portal and start brainstorming

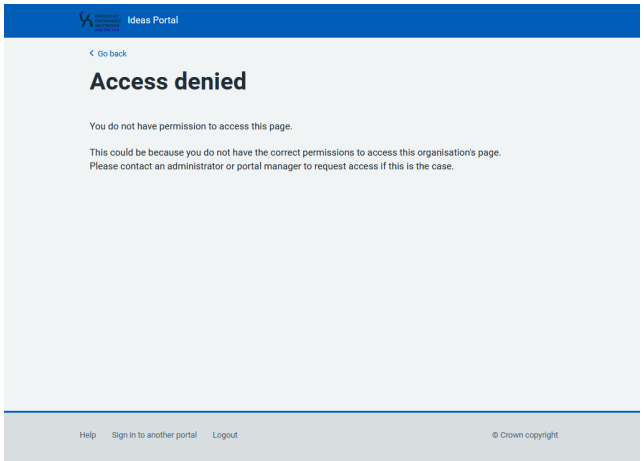

Figure A.10: Access Denied Page

A.3 Landing Pages

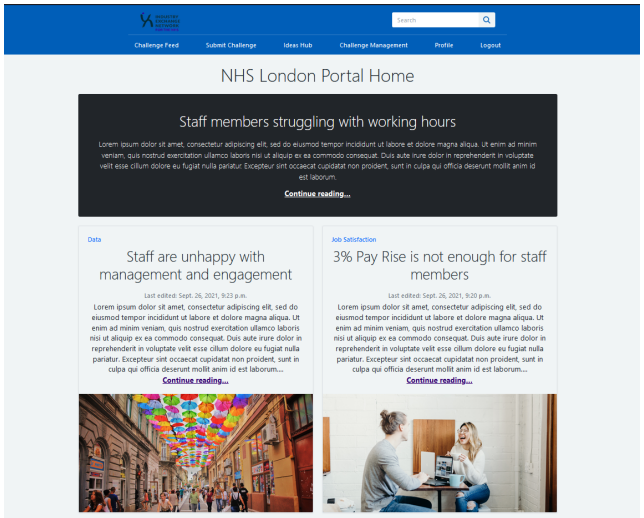

Figure A.11: Custom Landing Page

Once an account is registered users can login to their portal(s) of affiliation. Anonymous users can access any portal but are unable to submit challenges or ideas.

The landing page includes a summary of latest notifications for the logged in user, organisation wide statistics and a leaderboard to view top 5 'creators' in each portal.

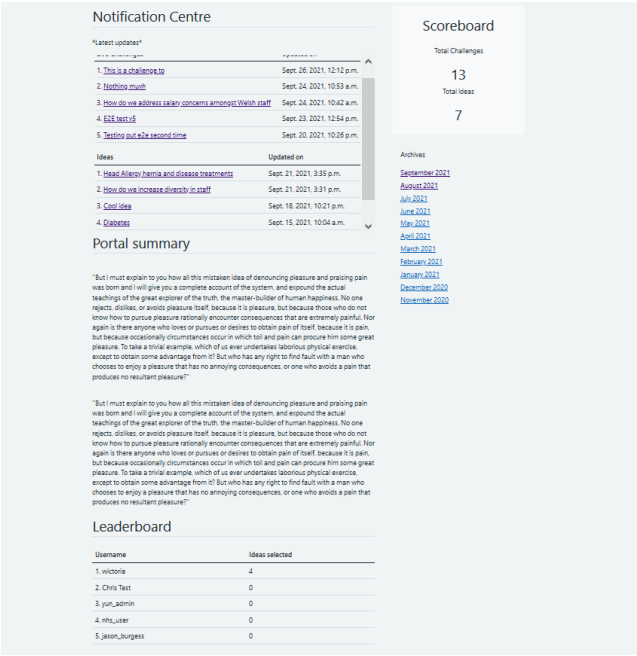

Figure A.12: Custom Landing Page 2

Admins and portal manager can customise the portal summary with information of their choosing and also select which polls users influence levels of engagement with certain challenges at - https://139.162.251.153/ideaportal-management

A.4 Components

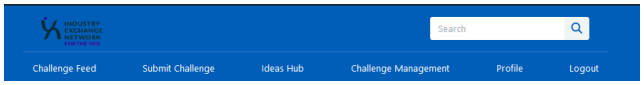

Figure A.13: Navigation bar for admins and challenge managers

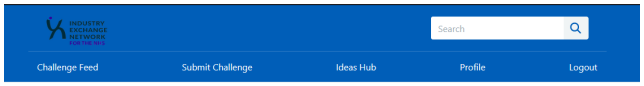

Figure A.14: Standard Navigation bar

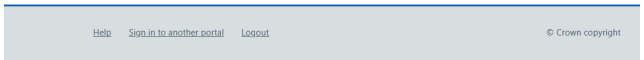

Figure A.15: Footer

For quick navigation around the portal to frequently accessed areas. The navigation varies depending on the permissions of the logged in user

A.5 Help Pages

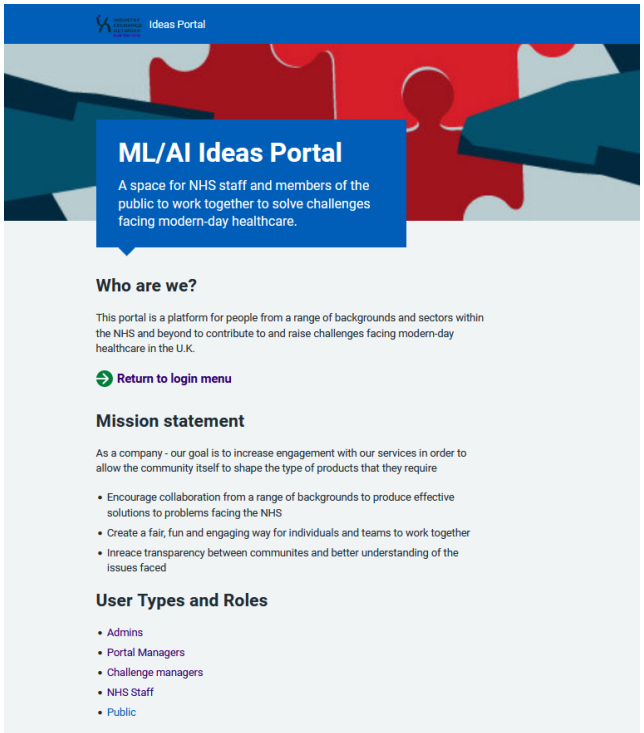

Figure A.16: Main Help Page

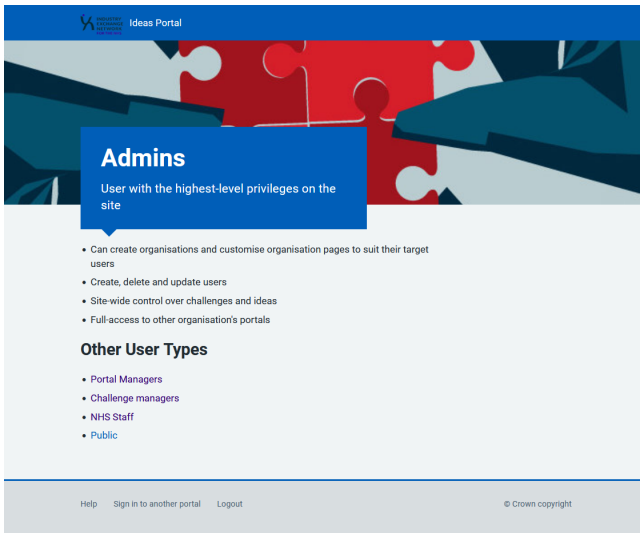

Figure A.17: Admin Info Page

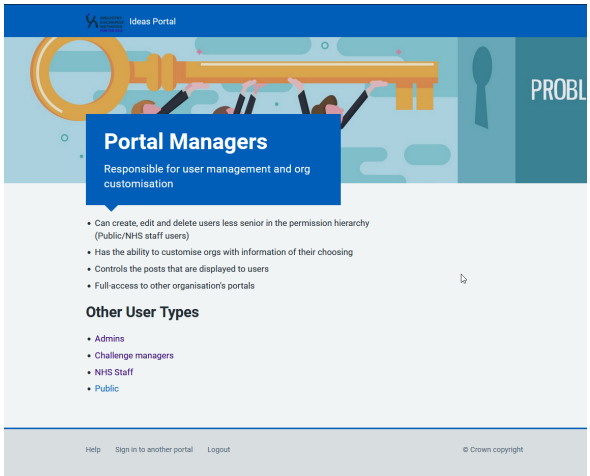

Figure A.18: Portal Manager Info Page

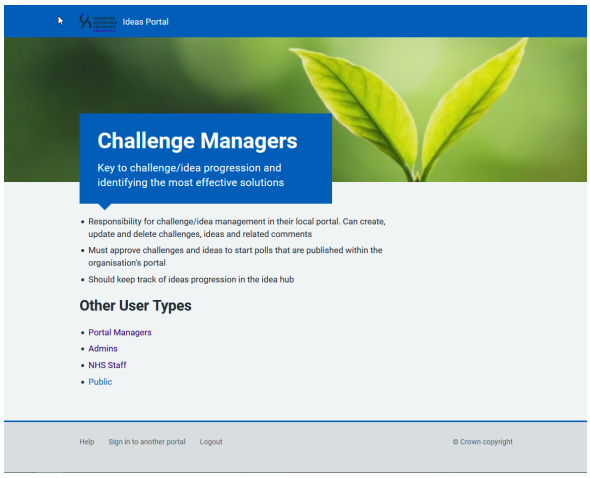

Figure A.19: Challenge Manager Info Page

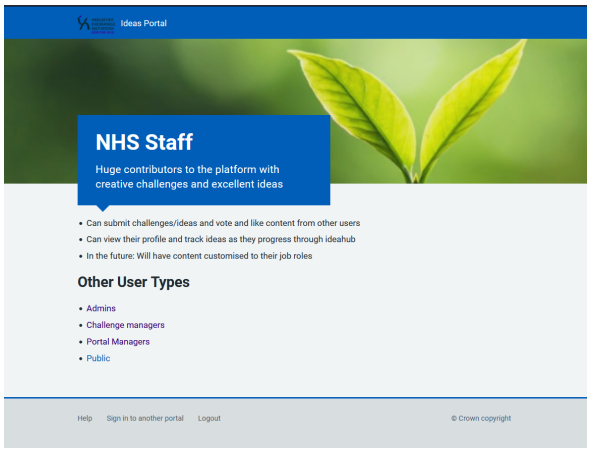

Figure A.20: NHS Staff Info Page

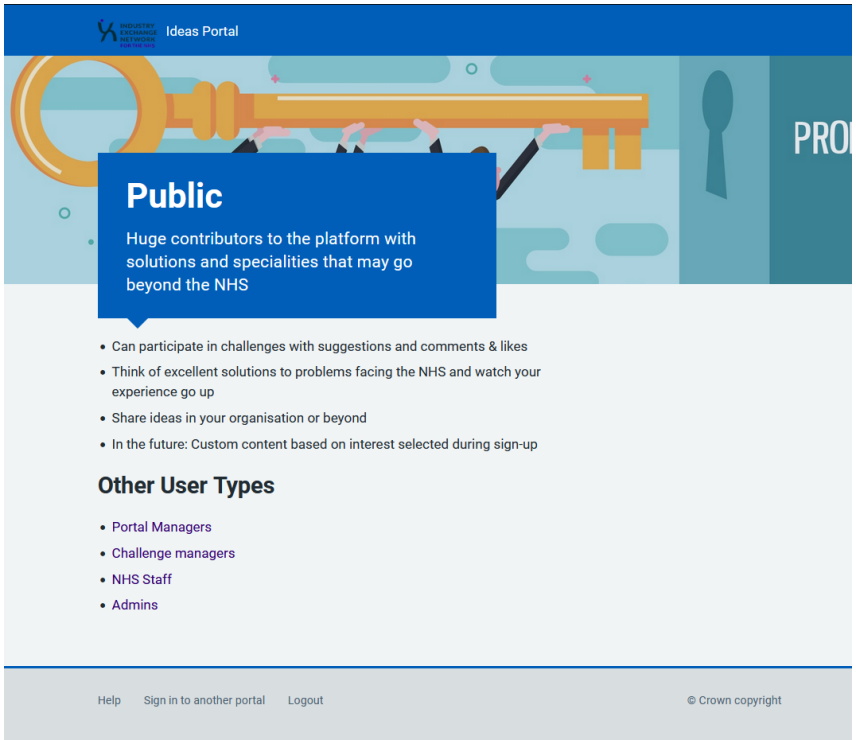

Figure A.21: Public User Info Page

A.6 Polls and Challenges

In the challenge feed users are able to view a list of polls ranked from the highest to lowest severity and categorised by the department.Users have the option to read more about a challenge or submit an idea with a help menu on the side to support.

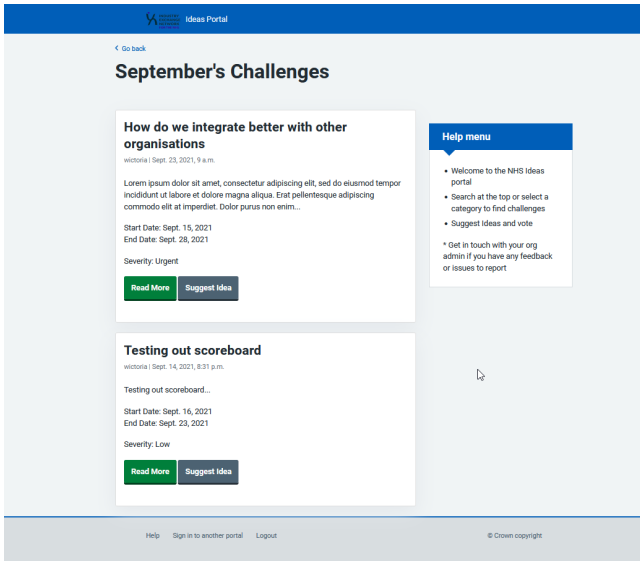

Figure A.22: Monthly Challenge Archives

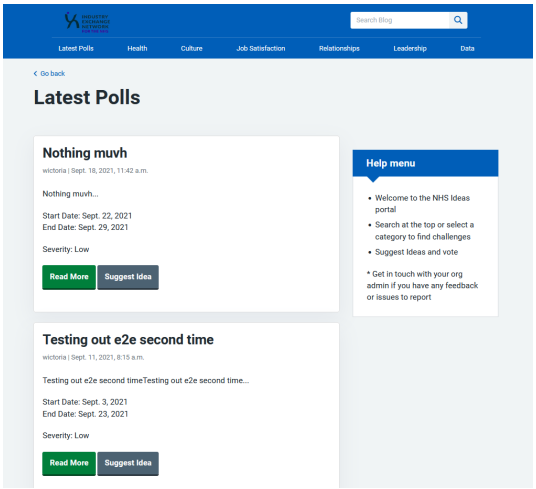

Figure A.23: Challenge Feed and Latest polls

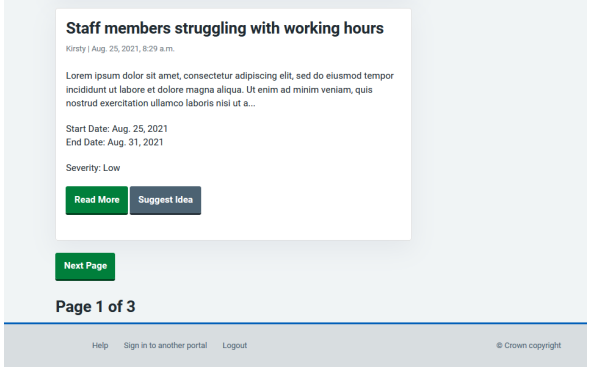

Figure A.24: Pagination example

If the user chooses the 'read more' option then they are able to upvote challenges to help them gain traction and visibility to other users on the site. Users are also able to comment and view existing comments in the side-bar (Figure A.25).

In addition to this, they can also suggest ideas directly from this page, view existing ideas and read more about them if they choose to do so. There is also a count of total ideas in a poll and a direct link to the idea with the most likes i.e. current winner (Figure A.26)

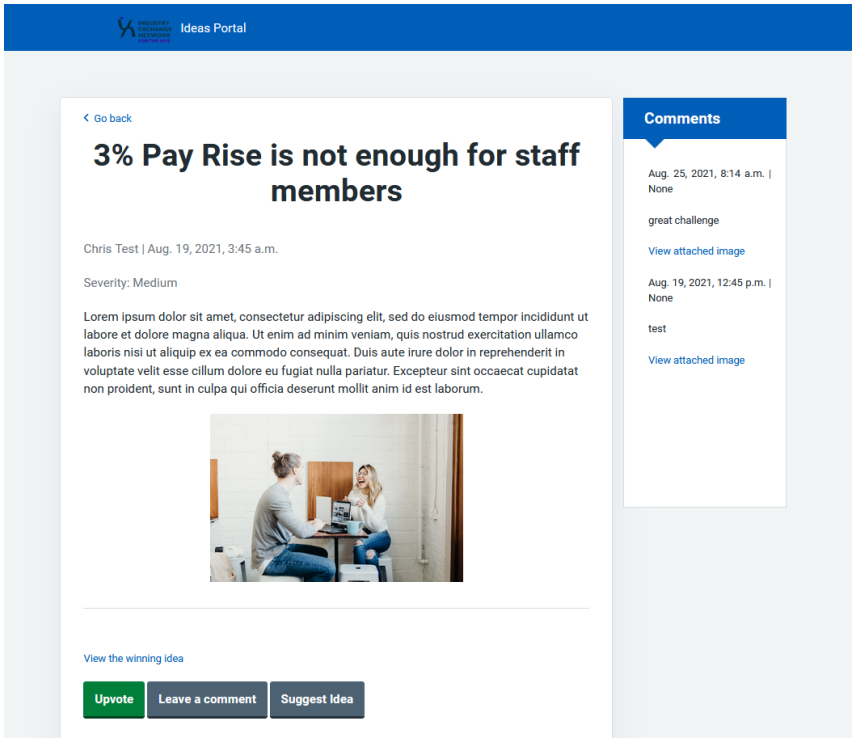

Figure A.25: Poll Detail Page

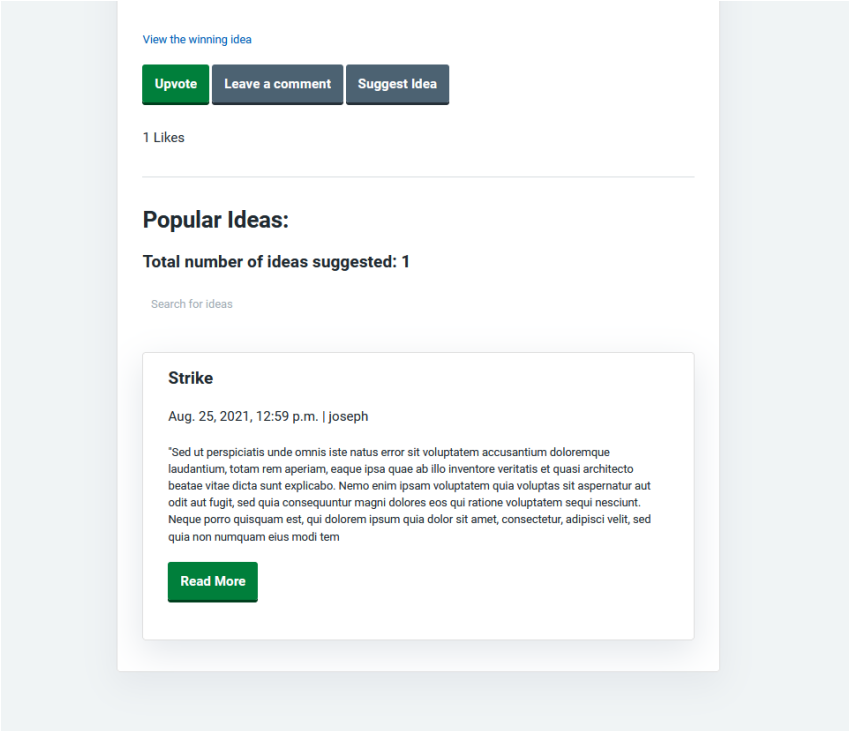

Figure A.26: Poll Detail Page 2

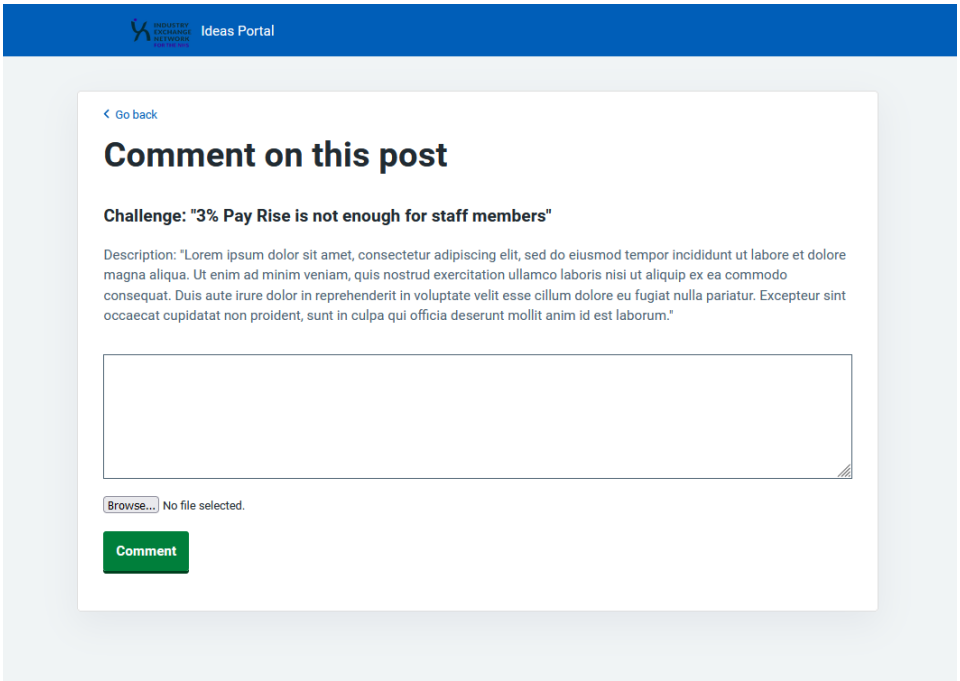

Figure A.27: Comment view

A.7 Ideas

Standard idea forms ask the user to fill a title and description as the only two mandatory fields (Figure A.28).

When a poll closes the idea with the most likes is automatically selected as the winner and the idea's stage updates from None to 'open'. At this point it is visible under the organisation banner in the Ideas Hub.

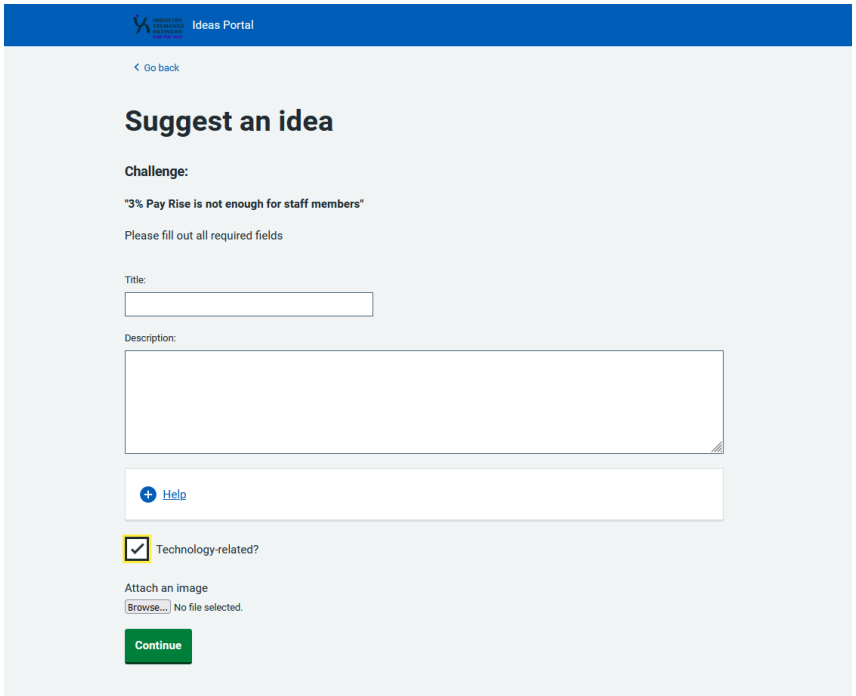

Figure A.28: Submit an idea Form

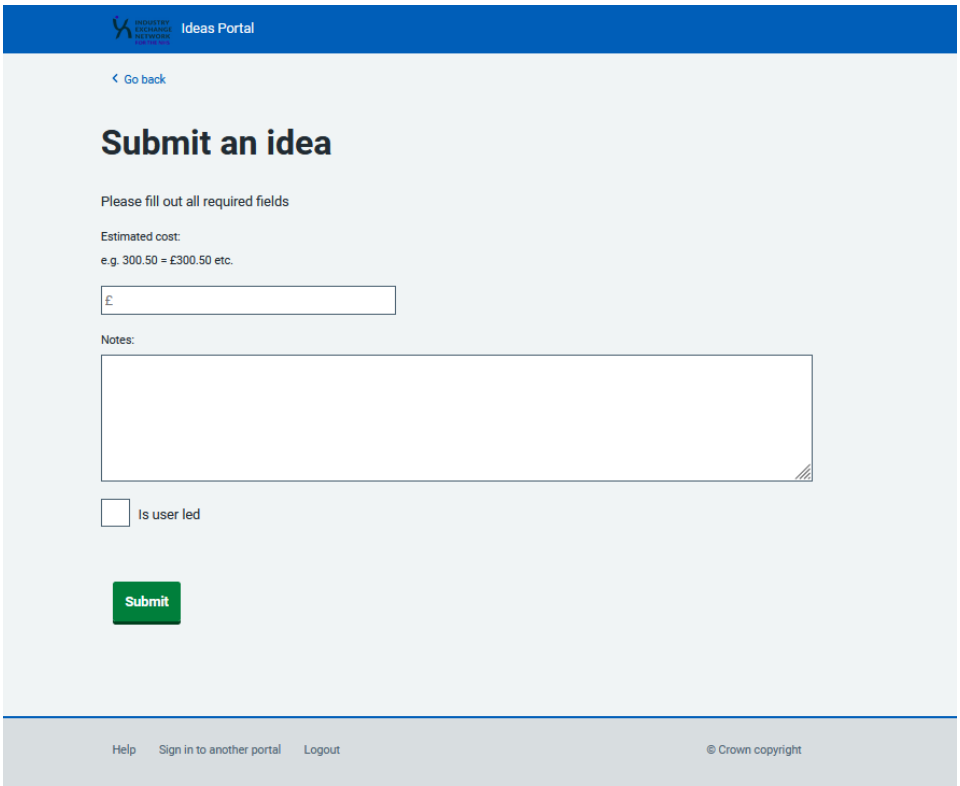

Figure A.29: Standard Idea Criteria Form

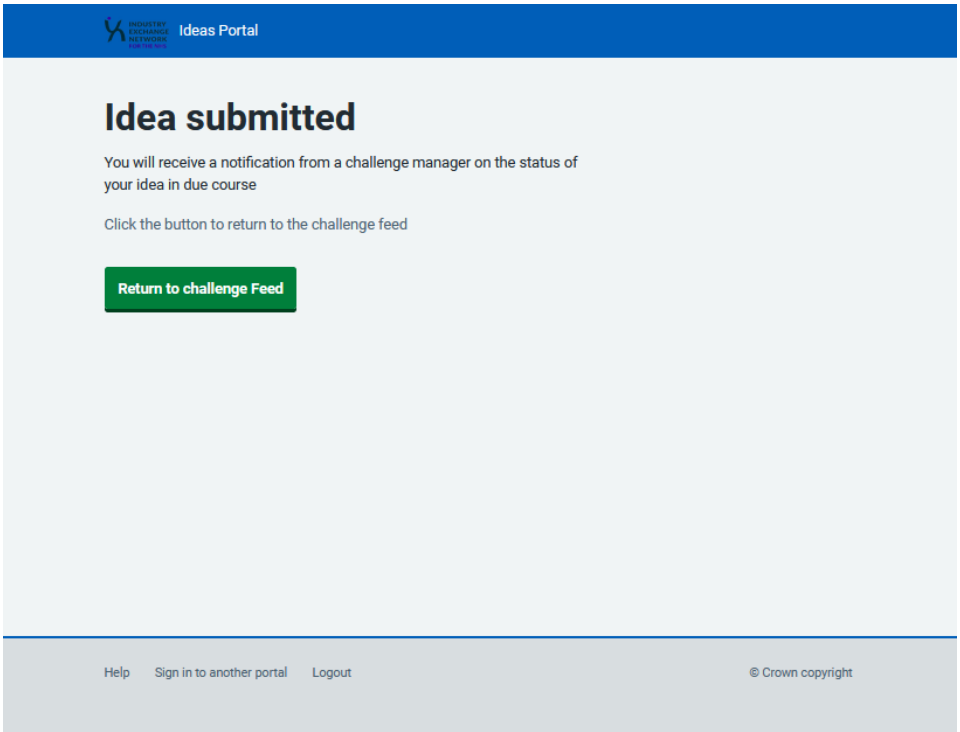

Figure A.30: Idea Submission Success Page

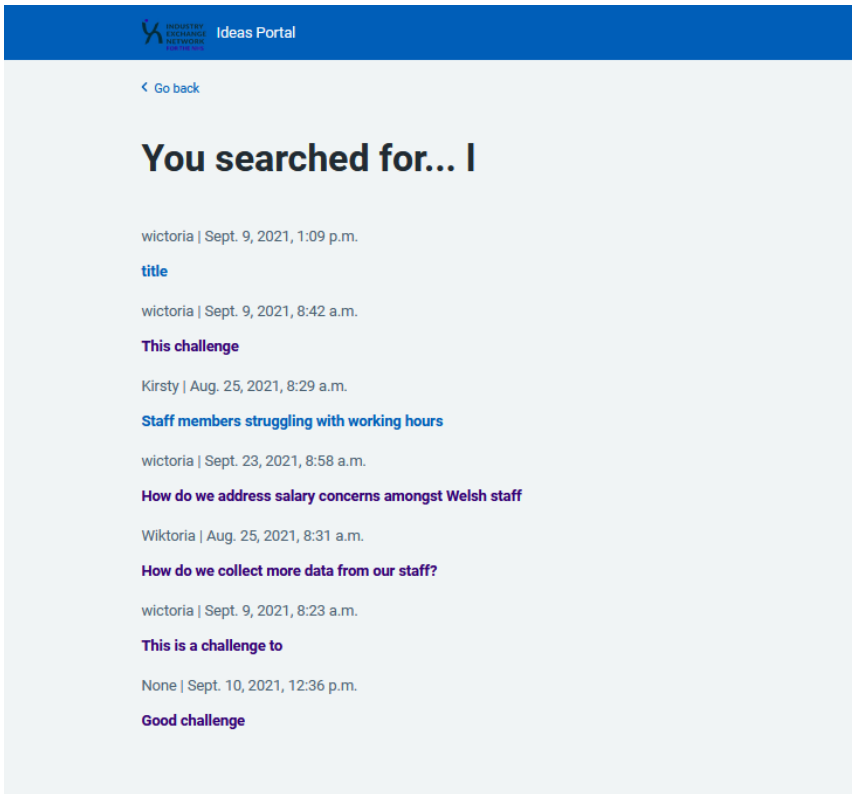

Figure A.31: Search Blog Page

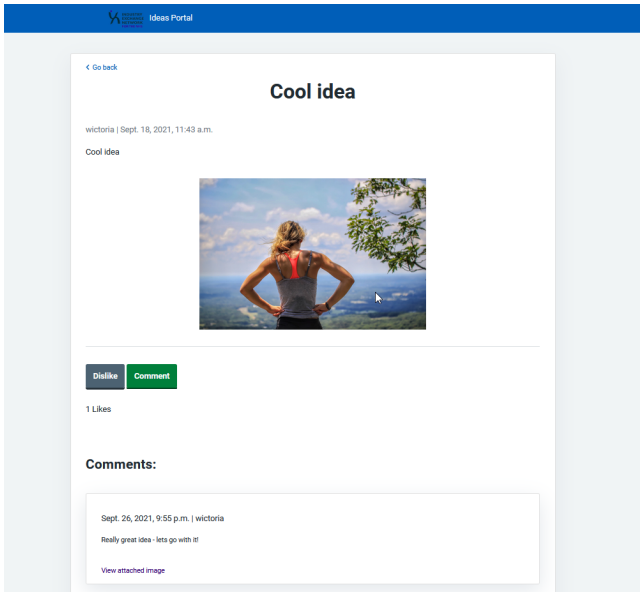

Figure A.32: Idea Detail Page

Portal managers and admins are able to specify whether they would like to activate customised org forms.This will influence the information their users fill out in idea forms.

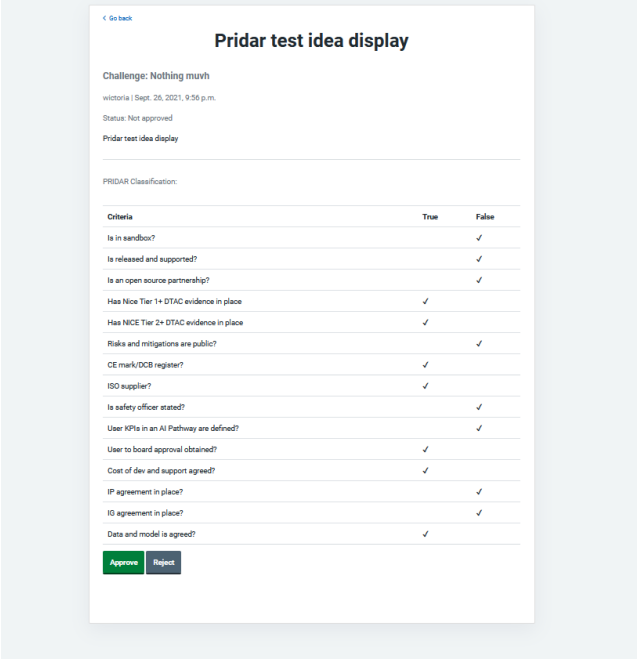

Figure A.33: Custom (PRIDAR) Idea Detail Page

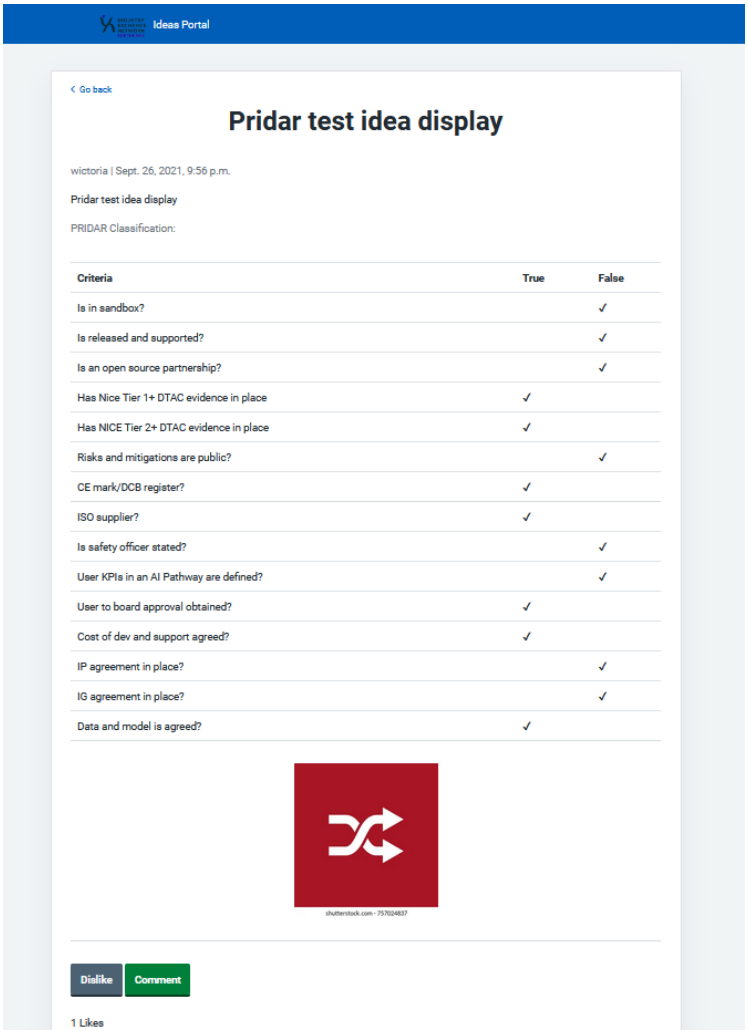

Figure A.34: Custom (PRIDAR) Idea Management Detail Page

If the a user selects the 'Technology-related' option when filling out an idea they will be asked to fill our the org-specific criteria - in this case the PRIDAR schema.

The management form then shows the to challenge managers the boolean criteria so that they can make a decision as to whether to publish the idea publicly so that users can engage with it or not.

A.8 Profile Page and Settings

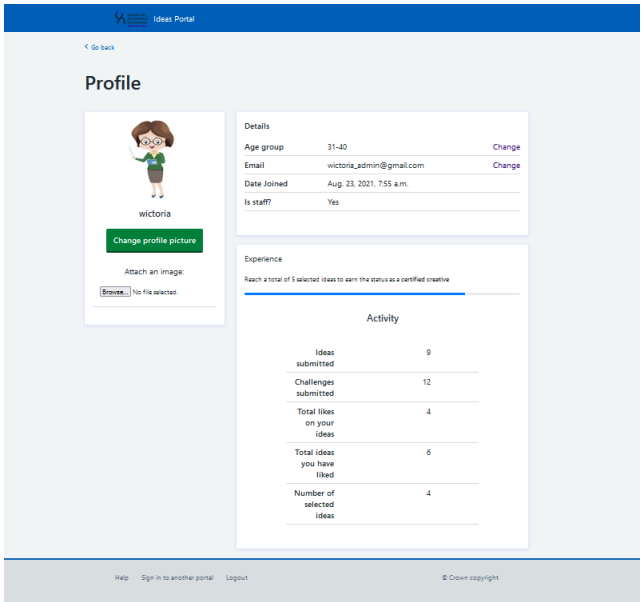

Figure A.35: Profile Page

On the profile page users are able to view their challenge and idea engagement statistics as well as their own personal details. They also have the option to change profile picture, email address or age group.

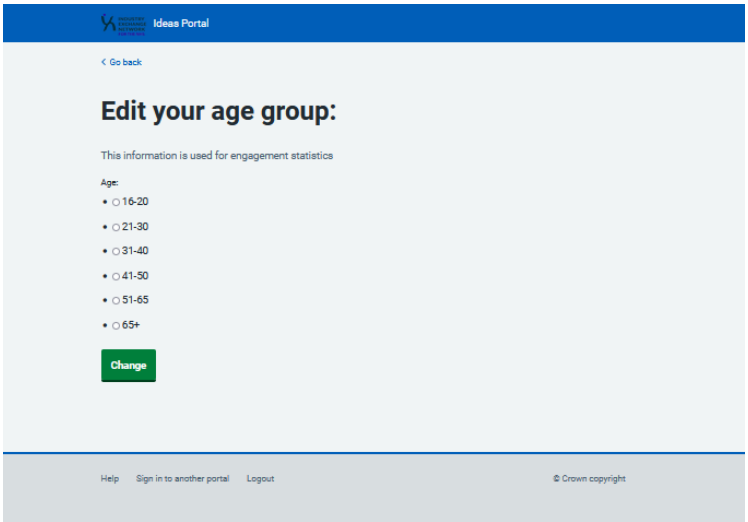

Figure A.36: Edit age form

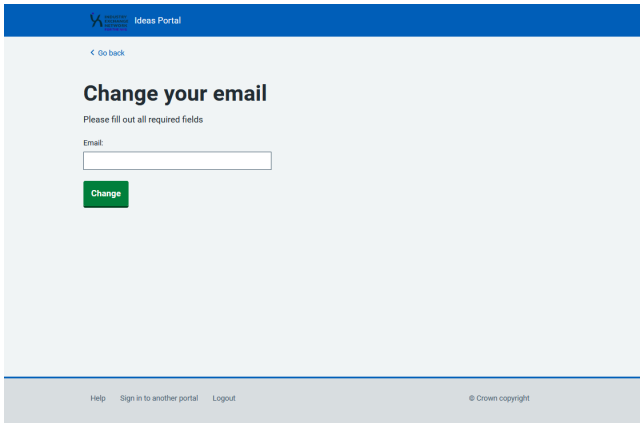

Figure A.37: Edit email form

A.9 Ideas Hub

In the ideas hub section of the portal users are able to track ideas in development. They can search for ideas using the search box or alternatively view ideas in progress by filtered by organisation.

Users can then see winning ideas and follow their transition from the initial 'open' stage to the 'delivered' stage. This section of the site is integrated with another platform called Trustnet from which users will be redirected to ideas in the portal. Members of staff will have the responsibility of updating idea stages via an API PATCH request and this will reflect in the UI of the idea life cycle.

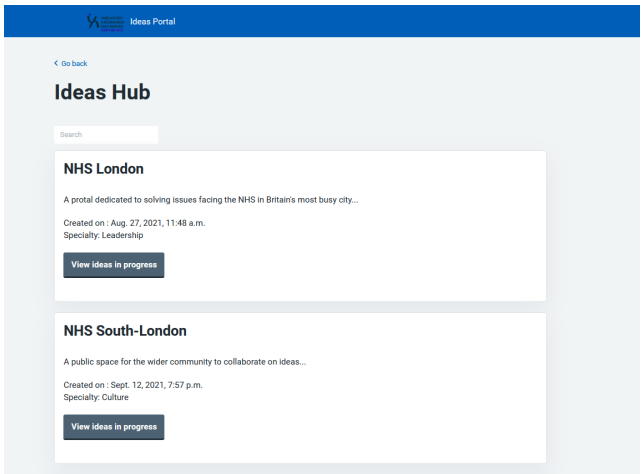

Figure A.38: Idea Hub search

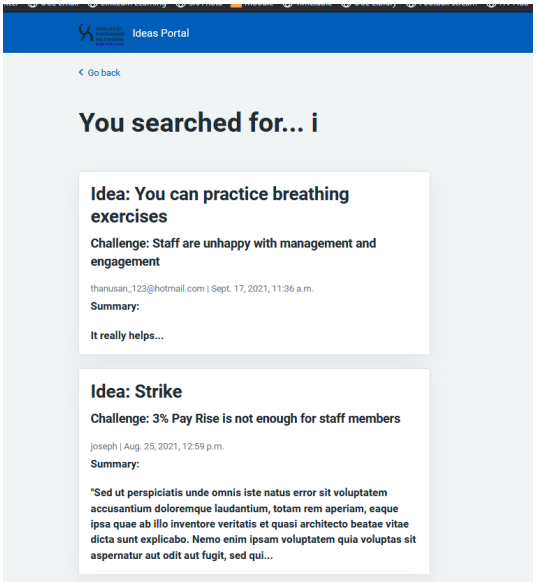

Figure A.39: Idea Hub search

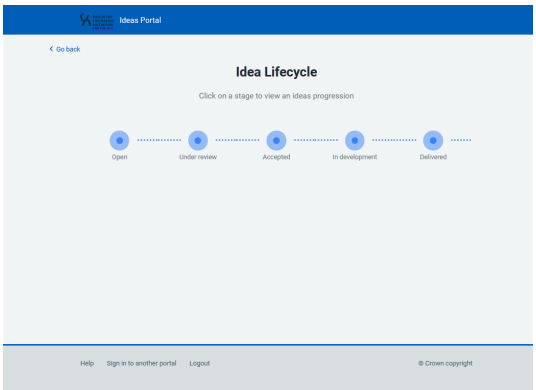

Figure A.40: Idea Lifecycle timeline

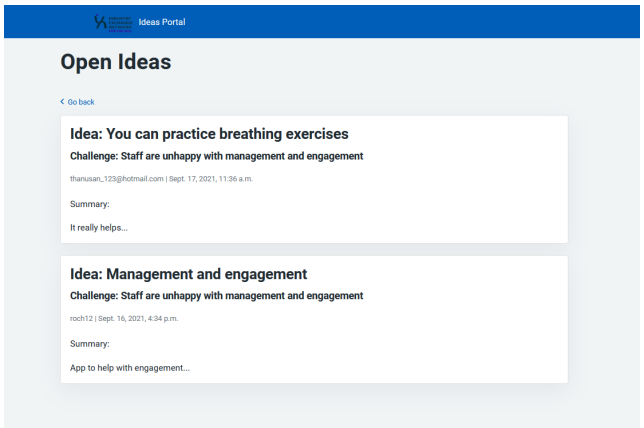

Figure A.41: Idea Stage view

A.10 Challenge Management

This section of the portal is only available to advanced users. Here they are able to see a management panel where they can approve pending challenges and start polls by entering a start and end date. These polls will then be published publicly and users with affiliation to the organisation are able to view it.

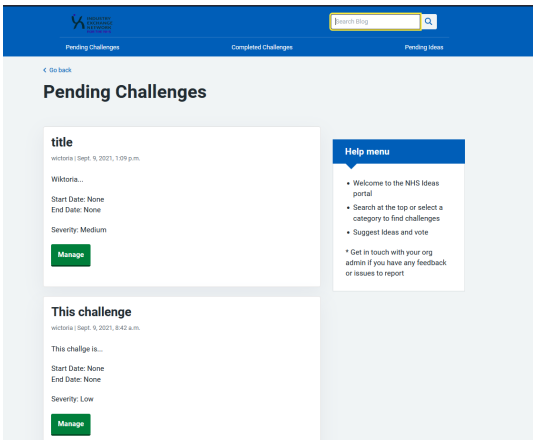

Figure A.42: Pending Challenges view

Advanced users are also able to view completed challenges and their winners for reference.

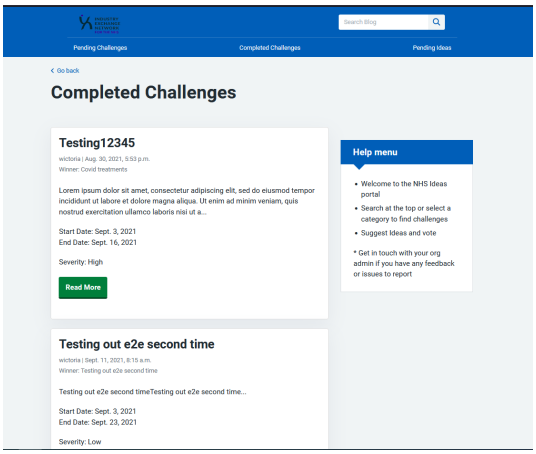

Figure A.43: Completed Challenges view

Finally, these user can approve pending ideas that they perceive as credible and potential solutions to challenges at hand.

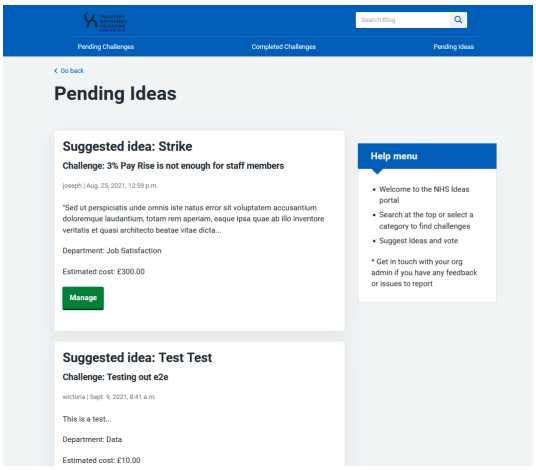

Figure A.44: Pending Idea view

Appendix B

Deployment Manual

1. Download the github repository or uploaded moodle zip file
2. Request a user for the Linode
3. Copy project folder to remote server using the command - 'scp views.py user@139.162.251.153: /'
4. Log in to server using the command - 'ssh user@139.162.251.153'
5. 'sudo -i -u postgres' to log into postgresSQL in Ubuntu/Linux terminal
6. Create a new PostgresSQL Database: Choose user name and password and then select apply
7. 'cd /home/user' to login into your user's directory on the server
8. 'source django-env/bin/activate' to activate the virtual environment
9. 'cd ideaportal-project' to change to the directory of the Ideaportal application
10. 'python manage.py makemigrations' to make sure database tables and columns are up to date
11. 'python manage.py migrate' to complete database configuration
12. ' python manage.py collectstatic' to transfer the latest static files

13. ’pkill gunicorn’ to kill any exisiting communication between gunicorn and the Django application

14. ’gunicorn -c conf/gunicorn-config.py ideaportal.wsgi’ to activate the configuration file and re-deploy the application on the server with the latest changes

Appendix C

Use Cases

Use Case	Register to a portal
ID	UC1
Brief Description	All users can create an account, making them a registered user and receive a verification email with a token
Primary Actors	NHS Staff Users, Public Users
Preconditions	Must have a valid email address

Main Flow	(a) The user clicks 'enter as a guest or register an account; (b) The user views the descriptions for each portal (c) The user clicks 'register' for a portal of their choice (d) The user fills out all mandatory information, including username, age group and email. (e) The system send a verification email to the user's registered email address. (f) The user clicks the token hyperlink in the email and is redirected to the 'Activation Success' page (g) The user signs into the portal they are affiliated with
Post Conditions	• The system saves the account details. • The user can now log into their affiliated portal with their username and password • The user cannot access any other portal and instead see an 'access denied' message
Alternative Flows	(a) The user enters the wrong credentials and an info is flagged up (b) The user resests their password

Table C.1: UC1 Register to a portal.

Use Case	Submit a challenge
ID	UC2
Brief Description	All users can submit a challenge for review.
Primary Actors	All users
Preconditions	User must have already been registered to the application
Main Flow	(a) The user logs in to any portal they are affiliated with (b) The user clicks the the 'submit challenge' tab in the navigation bar (c) The user fills out the mandatory fields of the challenge form (d) The form is saved in the database and submitted to management for review (e) The user is informed that the challenge is being reviewed and redirected to the landing page
Post Conditions	• A challenge is submitted to management for review • Challenge managers and admins are able to see the challenge in pending challenges section of the management panel • The user is able to track challenge review in the notification centre

Alternative Flows	(a) The challenge is a duplicate of an existing one and an info is displayed informing the user is that there can be no duplicates (b) The user does not field all mandatory fields then an info is displayed instructing the user which fields to fill before the form can be sent for review (c) The user is signed in as a guest and hence does not have permission to submit a challenge

Table C.2: UC2 Submit a challenge

Use Case	Submit an idea
ID	UC3
Brief Description	All users can submit an idea
Primary Actors	All users
Preconditions	User must have already logged into the application

Main Flow	(a) The user clicks the 'Challenge Feed' tab in the navigation bar (b) The user is able to see the latest polls ranked by urgency and also view polls by department (c) The clicks 'read more' and reviews the challenge information to see if they can come up with a solution (d) The user upvotes the challenge because they think its important and leaves a comment (e) The user clicks the 'submit an idea' button, fills mandatory fields and ticks the 'technology-related' checkbox (f) The user fills out appropriate org-specific (PRIDAR) information (g) ORCHA API validates the ideas and returns similar applications based of user input (h) The user chooses to submit the form and it is then saved and submitted to management for review
Post Conditions	• Challenge managers and admins are able to view the idea in the pending ideas section • The user is able to track idea review in the notification centre

Alternative Flows	(a) The user does not check 'technology-related' box and is instead presented with the standard idea form (b) The user does not fill all mandatory fields and an info is displayed instructing them on what fields must be filled (c) The idea is rejected automatically for being a duplicate of an existing one (either by title or description) (d) API validation is switched off for that organisation so the user is not presented with similar applications to their solution

Table C.3: UC3 Submit an idea.

Use Case	UC4 Review challenges and ideas.
ID	UC4
Brief Description	Challenge Managers and Admin user can review challenges and ideas
Primary Actors	Challenge Managers and Admins
Preconditions	User must have already logged into the application and are Challenge Managers or Admins

Main Flow	(a) The user clicks on the 'challenge management' tab in the navigation bar (b) The user is able to click 'read more' and view challenge information (c) A pop-up message is displayed to the user prior to approving or rejecting a challenge/idea (d) The user is able to approve the challenge and set a start/end date for the challenge and it is removed from the pending areas (e) The user is able to see the same functionality with ideas (f) The system directs the user back to the pending challenge/idea page
Post Conditions	• The challenge/idea is removed from the respective pages in the management panel • The system status of the challenge/idea is changed from draft to publish and is visible in the challenge feed
Alternative Flows	(a) N/A

Table C.4: UC4 Review challenges and ideas.

Use Case	Update Idea Stage
ID	UC5

Brief Description	Ideas stages are updated from REST API and changes reflect in ideas hub
Primary Actors	All users
Preconditions	User must have already logged into the application
Main Flow	(a) The user performs a GET request on the Ideaportal API to obtain a winning idea (b) The user uses the body of the request in a PATCH request to change the idea stage from open to 'under review' (c) The user logs into the main Idea portal site and clicks the 'ideas hub' tab in the navigation bar (d) The user navigates to the organisation's page and follows the timeline to the 'under review' section (e) The idea that was previously in open stage is now under review
Post Conditions	• The change to the idea's stage should be reflected in the idea hub • user is able to see the update in the notification centre too

Alternative Flows	(a) The user enters an invalid stage when making the API PATCH request and is presented with an info (b) The user enters the wrong credentials and is unable to authenticate the API

Table C.5: UC5 Update Idea Stage.

Appendix D

Code Listing

This next section of contain the most important excerpts of code from the application. It will cover python classes and functions in addition to HTML pages as well as models that were used to build database tables.

```
@unauthenticated_user
def enter(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('select_org')
        else:
            messages.info(request, 'Username OR password is incorrect')


    context = {}
    return render(request, 'userauth/enter_portal.html', context)
```

Listing D.1: Authenticate and sign in to portal function

```
def submit_challenge(request, slug):
    if request.user.is_authenticated:
        form = ChallengeForm()
        logging.info(slug)
        org = slug
        orgobject = Organisation.objects.get(slug=slug)

        if request.method == "POST":
            form = ChallengeForm(request.POST)

            if form.is_valid():
                form.author = request.user
                challenge = form.save()
                challenge.author = request.user
                challenge.org_tag = orgobject
                cnwl = False
                challenge = form.save()
                Post.objects.create(author=challenge.author, title=challenge.title, severity=challenge.severity, department=challenge.department,
                challenge=challenge, description=challenge.description, org_tag = challenge.org_tag, image = challenge.image)

                return redirect('submit_challenge_successful', slug=slug)


        context = {'challengeform': form, 'org': org, 'custom_on' : orgobject.custom_form_on, 'orgslug' : slug}

        return render(request,'challenges/submit_challenge.html', context)
    else:
        return redirect('access_denied')
```

Listing D.2: Submit challenge function

```
class ideaform(CreateView):
    model = Idea
    template_name = 'ideas/submit_ideas_form.html'
    form_class = IdeaForm


    def get_context_data(self, **kwargs):
        context = super(ideaform, self).get_context_data(**
    kwargs)
        obj = get_object_or_404(Post, pk=self.kwargs['pk'])
        context['object'] = context['post'] = obj
        challenge_choice = Post.objects.get(slug=self.kwargs[
    'slug'])
        challenge_slug = Post.objects.get(slug=self.kwargs['
    slug']).slug
        logging.info(challenge_slug)

        portal_choice = Organisation.objects.get(slug=self.
    kwargs['orgslug'])
        portal_slug = Organisation.objects.get(slug=self.
    kwargs['orgslug']).slug
        context['orgslug'] = portal_slug
        logging.info(portal_choice)
        logging.info(portal_slug)
        custom_on = portal_choice.custom_form_on
        logging.info(custom_on)
        is_pridar = False
        if custom_on:
            is_pridar = True
        has_access = False
        if self.request.user.is_authenticated:
            has_access = True
        context['has_access'] = has_access
        context['challenge'] = Post.objects.get(slug=
    challenge_slug)
        context['custom_on'] = is_pridar
```

```
31
32          context
33          return context
34
35      def form_valid(self, form):
36          form.instance.post_id = self.kwargs['pk']
37          return super().form_valid(form)
38
39      def get_success_url(self):
40          pk=self.kwargs['pk']
41          orgslug=self.kwargs['orgslug']
42          slug = self.kwargs['slug']
43
44          return reverse_lazy('criteria', kwargs={'orgslug':
    orgslug, 'pk': pk, 'slug':slug,})
```

Listing D.3: Submit idea class

```
1 class IdeaViewSet(viewsets.ModelViewSet):
2     queryset = Idea.objects.all().filter(stage__isnull=False)
    .order_by('title')
3     serializer_class = IdeaSerializer
4
5
6 class CNWLIdeaViewSet(viewsets.ModelViewSet):
7     cnwl_id = Organisation.objects.get(name = 'CNWL').id
8     queryset = Idea.objects.filter(org_tag = cnwl_id).filter(
    stage__isnull=False).order_by('title')
9     serializer_class = IdeaSerializer
```

Listing D.4: Django-REST API serializer classes

```
class PostDetail(generic.DetailView):
    model = Post
    template_name = 'blogs/post_detail.html'


    def get_context_data(self, *args, **kwargs):
        context = super(PostDetail, self).get_context_data()

        stuff = get_object_or_404(Post, id=self.kwargs['pk'])
        total_likes = stuff.total_likes()
        all_ideas = Idea.objects.filter(post=self.kwargs['pk'
    ]).order_by('-likes')
        context["total_ideas"] = all_ideas.count()


        try:
            winning_idea = all_ideas.order_by("likes")[:1]
            winning_idea_id = winning_idea.values('id')[0]['
    id']
            winning_idea_slug = winning_idea.values('slug')
    [0]['slug']
            logging.info(winning_idea_slug)
            context['winnerpk'] = winning_idea_id
            context['winnerslug'] = winning_idea_slug
            stuff.winner = Idea.objects.get(id =
    winning_idea_id)
            stuff.save()
            logging.info(stuff.winner)
        except:
            logging.info("fail")

        liked = False
        if stuff.likes.filter(id=self.request.user.id).exists
    ():
            liked = True

```

```
32        context["total_likes"] = total_likes
33        context["liked"] = liked
34        portal_choice = Organisation.objects.get(slug=self.
    kwargs['orgslug'])
35        portal_slug = Organisation.objects.get(slug=self.
    kwargs['orgslug']).slug
36        context['orgslug'] = portal_slug
37
38        return context
```

Listing D.5: Poll detail class

```
1
2 class IdeaDetail(generic.DetailView):
3     model = Idea
4     template_name = 'blogs/idea_detail.html'
5
6     def get_context_data(self, *args, **kwargs):
7         context = super(IdeaDetail, self).get_context_data()
8
9         stuff = get_object_or_404(Idea, id=self.kwargs['pk'])
10        idea_post = stuff.post.slug
11        idea_pk = stuff.post.id
12        idea_pridar = stuff.is_pridar
13        custom = False
14        if idea_pridar:
15            customised = OrgForm.objects.get(title = stuff.
    title)
16            custom = True
17            context['in_sandbox'] = customised.in_sandbox
18            context['is_released_and_supported'] = customised
    .is_released_and_supported
19            context['is_open_source_partnership'] =
    customised.is_open_source_partnership
20            context['NICE_Tier1_DTAC_evidence_in_place'] =
    customised.NICE_Tier1_DTAC_evidence_in_place
21            context['NICE_Tier2_DTAC_evidence_in_place'] =
    customised.NICE_Tier2_DTAC_evidence_in_place
22            context['risk_and_mitigations_are_public'] =
    customised.risk_and_mitigations_are_public
23            context['ce_mark_dcb_register'] = customised.
    ce_mark_dcb_register
24            context['safety_officer_stated'] = customised.
    safety_officer_stated
25            context['iso_supplier'] = customised.iso_supplier
26            context['user_kpis_is_an_ai_pathway_are_defined']
     = customised.user_kpis_is_an_ai_pathway_are_defined
27            context['user_to_board_approval_obtained'] =
```

```
customised.user_to_board_approval_obtained
28          context['cost_of_dev_and_support_agreed'] =
customised.cost_of_dev_and_support_agreed
29          context['ip_agreement_in_place'] = customised.
ip_agreement_in_place
30          context['ig_agreements_in_place'] = customised.
ig_agreements_in_place
31          context['data_and_model_agreed'] = customised.
data_and_model_agreed
32      context['custom'] = idea_pridar
33      logging.error(idea_post)
34      context['slug'] = idea_post
35      context['pk'] = idea_pk
36      logging.error(self.kwargs['pk'])
37      idea_comments = IdeaComment.objects.filter(idea=self.
kwargs['pk'])
38      logging.error(idea_comments)
39      context['idea_comments'] = idea_comments
40
41
42      total_likes = stuff.total_likes()
43
44
45      liked = False
46      if stuff.likes.filter(id=self.request.user.id).exists
():
47          liked = True
48
49      context["total_likes"] = total_likes
50      context["liked"] = liked
51      portal_choice = Organisation.objects.get(slug=self.
kwargs['orgslug'])
52      portal_slug = Organisation.objects.get(slug=self.
kwargs['orgslug']).slug
53      logging.error(portal_choice)
54      logging.error(portal_slug)
```

```
55        context['orgslug'] = portal_slug
56
57        return context
58
```

Listing D.6: PRIDAR idea detail class

```
<div class="row g-5">
    <div class="col-md-8">
        <p class="blog-post-title border-bottom">Notification Centre</p>
        <br>
        <p>*Latest updates*</p>
        <div class="scroll">
       <table class="table">
          <thead>
            <tr>
              <th>Live Challenges</th>
              <th>Updated on</th>

            </tr>
          </thead>
          <tbody>
            {% for item in notifications %}
            <tr>
              <td>{{ forloop.counter }}. <a href="{% url ’post_detail’ orgslug item.pk item.slug %}">{{ item.title }}</a></td>
              <td>{{ item.updated_on }}</td>
            </tr>

            {% endfor %}
          </tbody>
        </table>

        <table class="table">
          <thead>
            <tr>
              <th>Ideas</th>
              <th>Updated on</th>

            </tr>
```

```
34          </thead>
35          <tbody>
36            {% for item in idea_notifications %}
37            <tr>
38              <td>{{ forloop.counter }}. <a href="{% url '
    idea_post' orgslug item.pk item.slug %}">{{ item.title }}<
    /a></td>
39              <td>{{ item.updated_on }}</td>
40            </tr>
41
42            {% endfor %}
43          </tbody>
44        </table>
45        <p></p>
46 </div>
47
```

Listing D.7: Notification centre HTML

```
<center>
        <h1>{% block title %} {{ object.title }} {% endblock title %}</h1>
      </center>
        <p class=" text-muted">{{ post.author }} | {{ post.created_on }}</p>
        <p class=" text-muted"> Severity: {{ object.severity| capfirst }}</p>
        <p class="card-text ">{{ object.description | safe }}</p>

        {% if object.image %}
        <center>
          <img src="{{ object.image.url }}" style="max-width:50%;max-height:50%;">
          {% else %}
          {% endif %}
       </center>

        <hr>

        {% if total_ideas != 0 %}
        <br>
        <a href="{% url 'idea_post' orgslug winnerpk winnerslug %}">View the winning idea</a>
        <br><br>

                    <form action="{% url 'like_post' orgslug post.pk post.slug %}" method="post">
                        {% csrf_token %}
                        {% if user.is_authenticated %}

                        {% if liked %}
                        <button id="submit-button" type="submit" name="post_id" value="{{ post.id }}" class="nhsuk-
```

```
button nhsuk-button--secondary">Downvote</button>
                    {% else %}
                    <button id="submit-button" type="submit" name="post_id" value="{{ post.id }}" class="nhsuk-button">Upvote</button>
                    {% endif %}
                    {% else %}
                    <a href="{% url 'enter' %}"><p>Login</a> to like </p>
                    {% endif %}
                    <button formaction="{% url 'comment_post' orgslug post.pk post.slug %}" type="submit" name="post_id" value="{{ post.id }}" class="nhsuk-button nhsuk-button--secondary">Leave a comment</button>
                    <a href="{% url 'ideaform' orgslug post.pk post.slug %}" class="nhsuk-button nhsuk-button--secondary">Suggest Idea</a>

                    <p>{{ total_likes }} Likes</p>
                  </form>
                  <hr>

                  <h2>Popular Ideas:</h2>
                  <h3>Total number of ideas suggested: {{total_ideas}}</h3>


```

Listing D.8: Poll detail HTML

```
class Organisation(models.Model):
    name = models.CharField(max_length=32)
    slug = models.SlugField(max_length=200, null=True, default='none')
    is_active = models.BooleanField(max_length=50, null=True, blank=True)
    specialty = models.ForeignKey(Department, on_delete=models.SET_NULL, related_name='specialty', null=True, blank=True)
    created_on =  models.DateTimeField(auto_now_add=True)
    description = models.CharField(max_length=500, null=True, blank=True)
    image = models.ImageField(null=True, blank=True,)
    post1 = models.ForeignKey("blog.Post", on_delete=models.SET_NULL, related_name='postorgtag1', null=True, blank=True)
    post2 = models.ForeignKey("blog.Post", on_delete=models.SET_NULL, related_name='postorgtag2', null=True, blank=True)
    post3 = models.ForeignKey("blog.Post", on_delete=models.SET_NULL, related_name='postorgtag3', null=True, blank=True)
    default_pic_mapping = { 'Health': random.choice(health), 'Culture': random.choice(culture), 'Job Satisfaction': random.choice(job_satisfaction),'Relationships': random.choice(relationships), 'Leadership': random.choice(leadership)}
    api_on = models.BooleanField(max_length=50,  default=True)
    custom_form_on = models.BooleanField(max_length=50, default=False)


    OPTIONS = [
    ('Health', 'Health'),
    ('Culture', 'Culture'),
```

```
20     ('Job Satisfaction', 'Job Satisfaction'),
21     ('Relationships', 'Relationships'),
22     ('Leadership', 'Leadership'),
23     ('Data', 'Data'),
24     ('Other', 'Other')
25     ]
26     section1 = models.CharField(max_length=1000, null=True,
    blank=True)
27     section2 = models.CharField(max_length=1000, null=True,
    blank=True)
28     section3 = models.CharField(max_length=1000, null=True,
    blank=True)
29     about = models.CharField(max_length=1000, null=True,
    blank=True)
30
31     def get_profile_pic_url(self):
32         if not self.image:
33             return self.default_pic_mapping[self.specialty.
    department]
34         return self.image
35
36     def save(self, *args, **kwargs):
37         self.slug = slugify(self.name)
38         self.image = self.get_profile_pic_url()
39         super(Organisation, self).save(*args, **kwargs)
40
41     def __str__(self):
42         return self.name
```

Listing D.9: Organisation model class

```
class ChallengeForm(forms.ModelForm):

    additional_field = forms.CharField(max_length=30)

    def __init__(self, *args, **kwargs):
        super(ChallengeForm, self).__init__(*args, **kwargs)
        self.fields['department'].queryset = Department.
    objects.filter(is_approved=True)
        self.fields['additional_field'].required = False
        self.fields['image'].required = False

    class Meta:
        model = Challenge
        fields = ['title','department', 'description', '
    severity', 'additional_field', 'image']
        labels = {'title':'Title', 'department':'Department',
     'description': 'Description', 'severity': 'Severity', '
    additional_field': 'Extra', 'image':'Add an attachment'}
```

Listing D.10: Challenge form class

Appendix E

Detailed Gantt Chart

Gantt chart is visible at full A4 size on the following page.

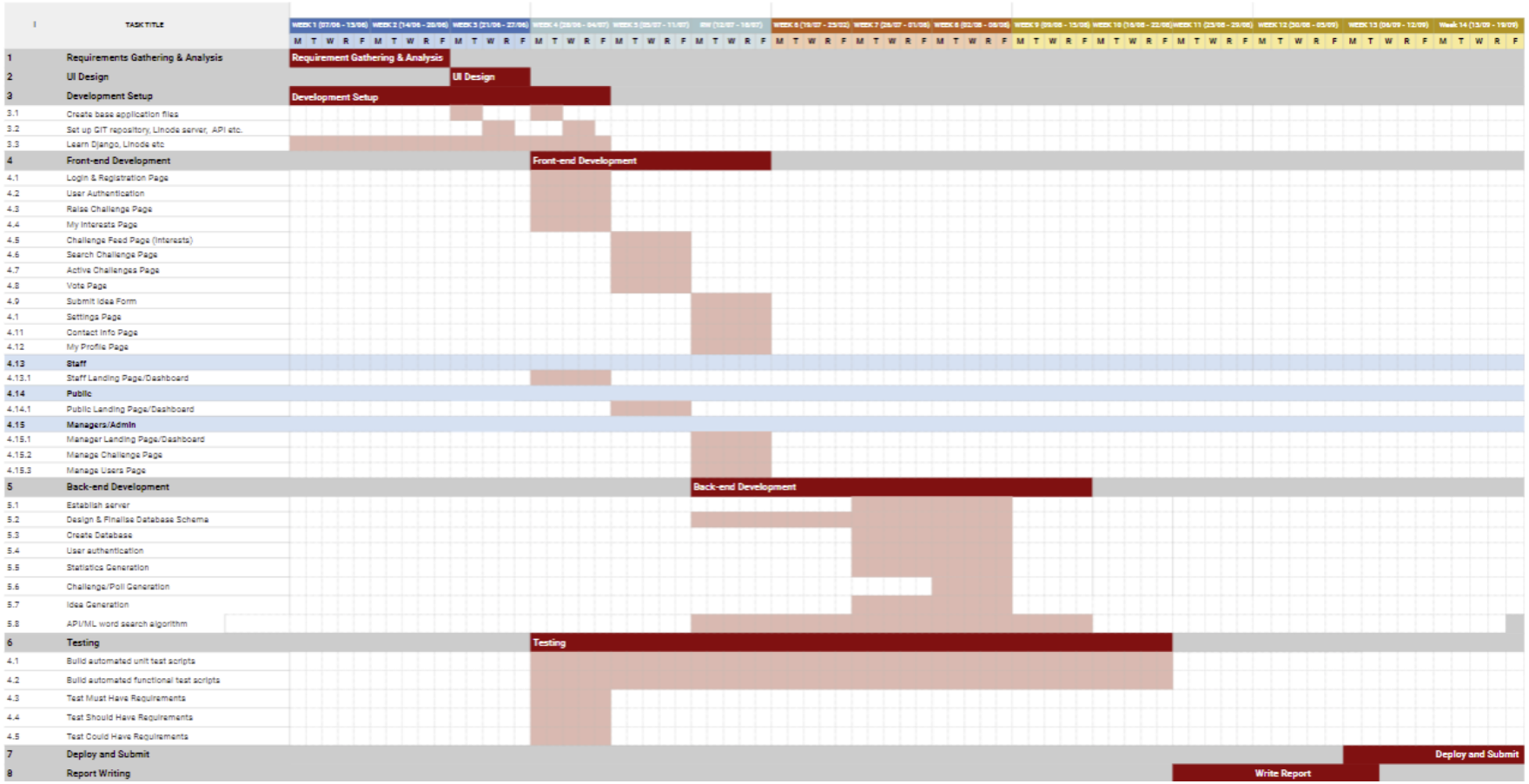

Figure E.1: Detailed Gantt Chart