

Introduction

Purpose:

Our Initial Value Proposition (IVP) is that business, researchers, Post-Secondary Institutions (PSIs), private labs and government groups (funders, labs etc) have no centralized, well-developed method of matching solution providers with solution seekers.

Scope:

The platform will contain data on resources, (talent, space, equipment, software and hardware solutions / products) which will be matched with well-defined projects needing a solution.

System Requirements

Functional Requirements:

To support the user stories, our application has the following requirements. Others may exist, but we've identified and organized as many as possible here. This section summarizes the key components.

SP.FR.001 Navigation

The application must be able to provide navigation as described in the Application Design section above or in the design specification document created by the designer.

SP.FR.002 Login

Users must log in to the application using an email address and password. Password requirements should follow as reasonably possible the current best practices the Open Web Application Security Project ([OWASP](#)) specifies.

SP.FR.003 Forgot Password Process

Users who forget their passwords can recover it using the [side channel token methodology](#) (link via email). For the purposes of the MVP we can skip the first step of identity confirmation using security questions as detailed in the specification linked above.

SP.FR.004 User Profile Editing

Logged-in users will be able to edit the content of their user profiles

- Name

- First name
- Last Name
- email address
- Office phone number
 - Field must allow for all different telephone number types to include country codes - for example +011, +1, etc
- Education
 - More than one field as some users may want to add several degrees to their profile
- Position (and any role-relative specifics)
- Languages spoken
- Experience Level (Junior / Senior / Expert)
- Categories of experience (as with profile creation on work-share communities, user will be typing and will be able to select from proffered choices) - industry classifications to be derived from [Standard Industrial Classifications](https://en.wikipedia.org/wiki/Standard_Industrial_Classification)
https://en.wikipedia.org/wiki/Standard_Industrial_Classification

SP.FR.005 Application Design

Elements of application design will be as shown in the wireframes or the design specification document. We will require a responsive design layout. We intend to support desktop, laptop and tablets.

SP.FR.006 User Administration Dashboard (Admin)

The technical nature of the platform and its' user match-making process requires some level of admin oversight. This process needs a user administration dashboard to input user details and initiate the invitation process.

Non Functional Requirements:

Performance :- The website's load time should not be more than one second for users.

Reliability :- Applicants can be able to access the website without failure.

Availability :- In case of unplanned system downtime, all features will be available after one working day.



Recoverability :- If the major incident happens on the website, it must take measures to go back to being fully operational within three days.

Usability :- The website's interface has to be user friendly and easy to use.

System Requirement

GCP(Google Cloud Platform) :-

Machine configuration

Machine type	e2-medium
CPU platform	Intel Broadwell
Architecture	x86/64
vCPUs to core ratio 	—
Custom visible cores 	—
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None

Networking

Public DNS PTR Record	None
Total egress bandwidth tier	—
NIC type	—

Boot disk

Image	Interface type	Size (GB)	Device name	Type	Architecture	Encryption	Mode	When deleting instance
debian-11-bullseye-v20220621	SCSI	10	frontend	Balanced persistent disk	x86/64	Google-managed	Boot, read/write	Delete disk

There are 3 VM's with the same above given configurations :-

- i. Dev
- ii. Alpha
- iii. Beta

Cloudflare :-

To use Cloudflare, you need to point the name servers for your domain to Cloudflare's ones.

The traffic is forwarded to the Cloudflare network, where it gets automatically filtered o prevent malicious traffic.

The name servers can be changed where you have registered the domain.

Cloudflare Name Servers:

NS ara.ns.cloudflare.com

NS chris.ns.cloudflare.com

DNS Record:

CNAME - www - thesciencepark.dev - Proxied - TTL(Auto)

A - thesciencepark.dev - 68.183.192.104 - Proxied - TTL(Auto)

A - alpha - 34.69.195.100 - Proxied - TTL(Auto)

A - beta - 34.172.105.190 - Proxied - TTL(Auto)

A - dev - 35.184.8.213 - Proxied - TTL(Auto)

SSL/TLS encryption mode is Flexible (Encrypts traffic between the browser and Cloudflare)

Alberta Science Park

Software Design Document

Version 1.1

Introduction

Document Purpose

The purpose of this document is to outline the technical aspects of the Alberta Science Park Web Application and the technologies used to develop and implement the application. The goal of this document is to give the reader a better understanding of how the application is being developed and implemented through examples of requirements, constraints, and system architecture.

Architectural Overview

Application overview

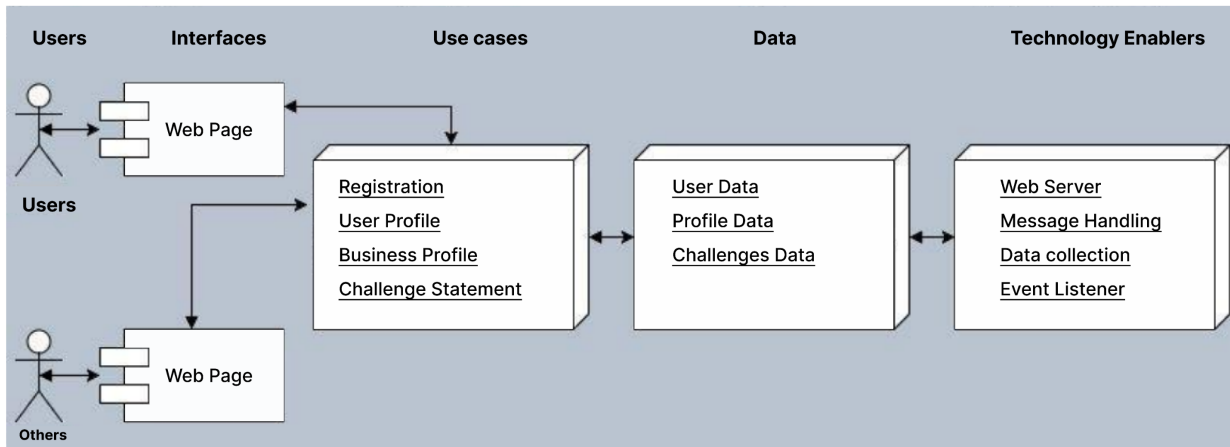


Figure 1: Application Overview and Diagram

Core application uses

Figure 1 highlights the use-cases for the Alberta Science Park application. These uses are the expectations of information to be presented to the user.

- Persona One - Geroge
 - His purpose for using the platform is to seek solutions to his particular engineering challenges which affect pipeline operations
- Persona Two - Graham
 - His purpose for using the platform is to ensure utilization of all the lab's resources, to help secure ongoing funding and attract a steady stream of research applicants to the lab.
- Persona Three - Nancy
 - Her purpose for using the platform is to be more efficient and effective in her sales process, by utilizing the matchmaking process between solution seeker and solution provider, by ensuring that the lab's CV is always up to date in the platform.

- **Persona Four - Brenda**
 - Her purpose for using the platform, as a platform owner, will be to ensure that the funds were well allocated and spent, and that the utility of the platform is not skewed toward one user group, but offers equally useful functionality to all users and user groups

Data

The core information and data constructs required to realize the core application uses are highlighted in Figure 1. The follow data is fundamental to the core use cases.

- **Users**
 - The current user data will be the current/potential users using the Alberta IoT application. This system will manage the propagation of user data from web services and local hardware.
- **Personal Information**
 - Personal Information data will be mapped against the particular users collecting their personal data viz. First name, Last name, Education, Position, Languages spoken, Experience Level, Address, Location etc.
- **Business Information**
 - The Business Information data will be stored as a business profile for each and every user. The data stored will be like Company Name, Website, Description, Office Phone, Company Classification, Company location etc.
- **Challenge Statement**
 - The challenge statement data will allow for permission based action between the web application as well as the web service. There will be a user to access the specific challenges he/she has created and also a common list of data to checkout all the challenges going on the platform.

Technology Enablers

Figure 1 highlights the key set of components to support implementation of the Alberta Science Park application.

- Web Server
 - The application will reside on a web server. The web server will be required for providing access to the Alberta Science Park IoT application interface and the REST api.
- Data Collection
 - The data collection will be implemented on a gateway machine that will propagate user and challenges data from the portal. This module will also be responsible for sending data to the web database through the REST api.
- Event Listeners
 - This module is the main function of the REST api. It will handle incoming events and request made over http from clients and web service users. It will then hand off the messages to the message handler when implemented on the portal in near future.

The Alberta Science park IoT application is built predominantly upon the Django framework. The Django framework uses a 3-tier architecture very similar to the popular Model, View, Controller (MVC) architecture. Figure 2 can be seen below and is a diagram of the implemented framework in respect to the web application. In Figure 2 below you can see that a Django project can have multiple applications within itself, and that the Alberta Science park IoT System is using two applications in tandem.

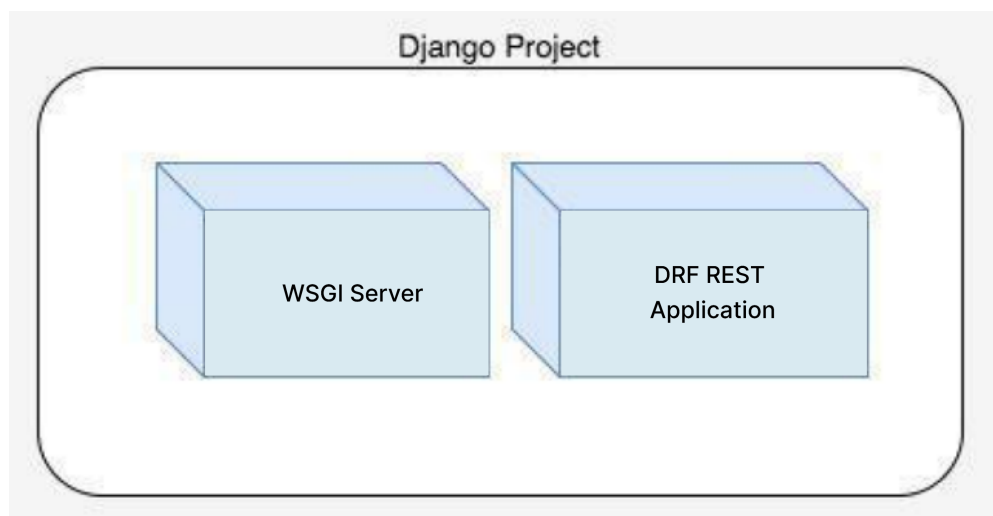


Figure 2: Web Application Layers Diagram

In Figure 2 above you see that there are two applications within the single Django Project. These two applications work together to accomplish the goals of the Alberta Science park IoT System. The Alberta Science park IoT Application is the application that is built upon the standard Django framework and is modeled after an MVC architecture.

Figure 3 below is a diagram of the Rest application layers. In the diagram below you can see that there are three main layers: Models, Views, and Controllers. Inside the Models you can see that there are the actual data objects that our application uses to store and display, as well as the REST Serializers that are able to serialize our data into JSON and make it easily consumable by the REST api. The Controller layer handles the GET/POST requests made by users and our application to send and receive data. The View layer is the piece of the application that allows our webpage to be created and displayed with data from the models.

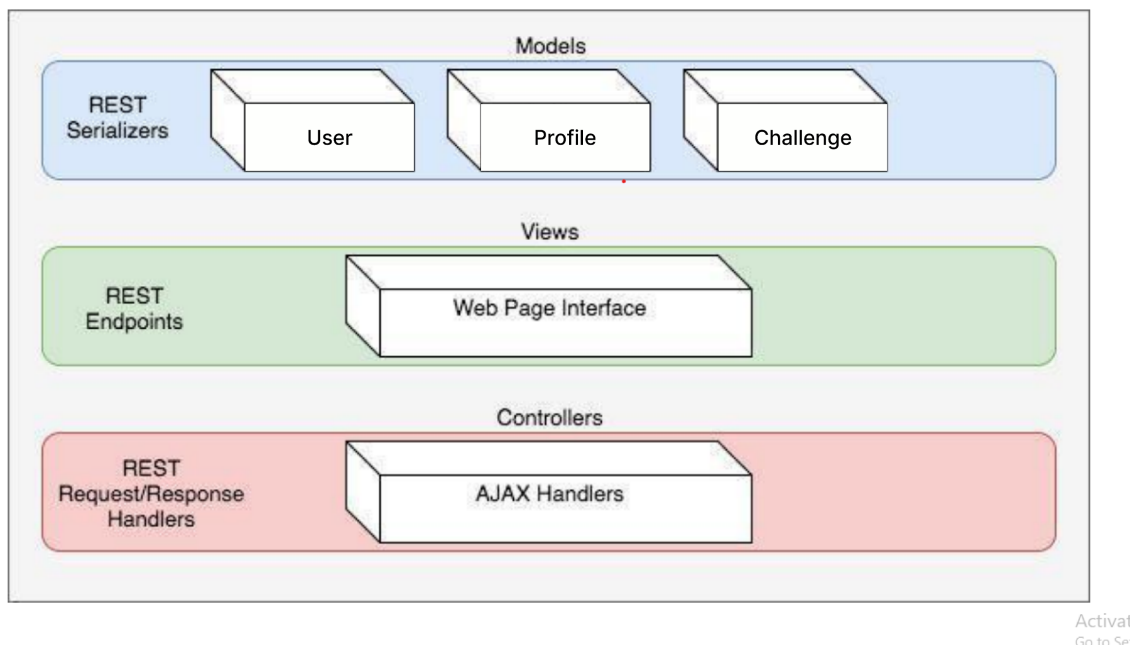


Figure 3: Layered View of Alberta Science Park Architecture

The Alberta Science park IoT application will use a fusion of two architectures Model, View, Controller architecture, and RESTful architecture. Inside our Django project you may recall that we have created two separate applications. The MVC will be used in the implementation of the web application interface and the RESTful architecture will be used for the backend, which will handle http request to api endpoints including ajax request from the REACT web application and external request from data collecting gateway machines. An overview of how the RESTful architecture works with our standard Django MVC architecture can be seen below in Figure 4.

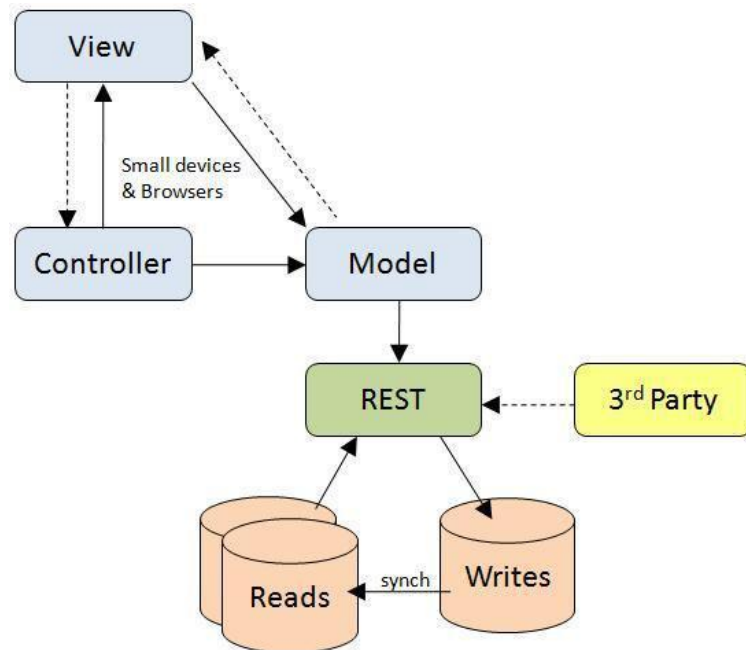


Figure 4: REST & MVC Architecture Diagram

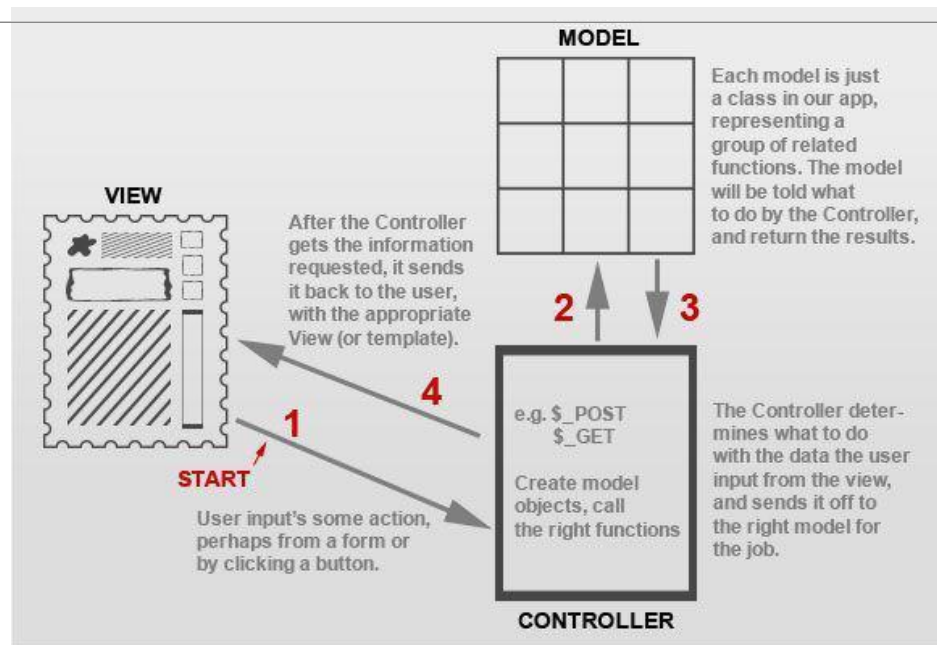


Figure 5: MVC Diagram

Module and Interface Description

Django

MVC framework vs Django MTV framework.

Our project will be using django at its core. This allows us to set up the website using a commonly used MVC framework. The MVC framework consists of 3 main components; Model, View, and Controller. The model portion of the framework consists of all of the classes that we will need for the project. The View is basically what will appear on the webpage. The controller is what links the Model and the View together.

In Django the underlying MVC architecture is actually slightly different from the classic MVC approach. Models are still Models in Django, but a View is actually called a Template, and a Controller is actually called a View. This means that a Django Template is actually what you see on the webpage, and a View links the classes in a Model together with a React Frontend Technology.

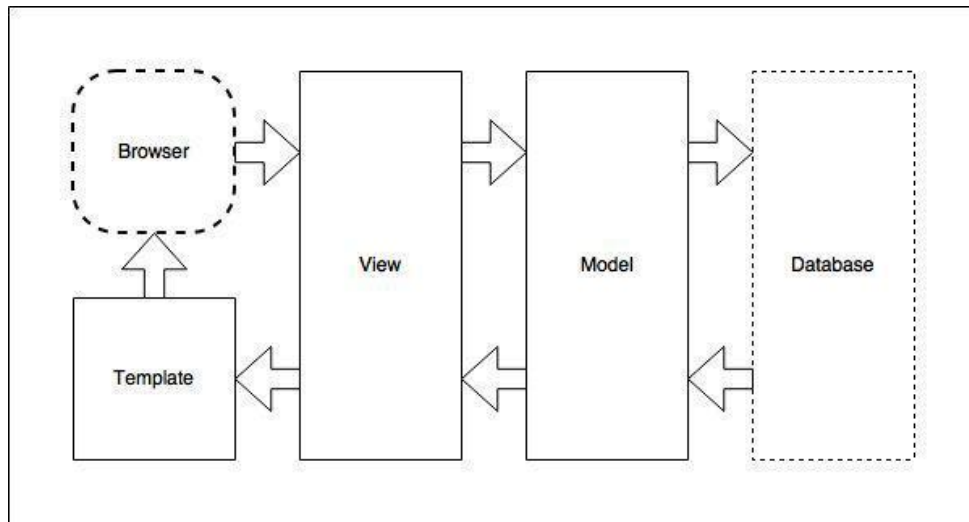


Figure 6.1: Django MTV Diagram

Flow of data in a Django MTV framework.

Figure 6 shown above showcases a brief overview of the flow of data in a basic Django application, beginning with a request from a browser and resulting in a web page produced back to the browser.

When a request is made to view a webpage provided by a Django application, it is first referenced in a list of url patterns located in a file “url.py”. The url patterns in this file will link directly to the View portion of the MTV framework by accessing a file called “views.py”

The file “view.py” basically holds all of the functionality for the Django application (which explains why we reference these as “controllers”), and uses the classes defined in your Model to manipulate the data before sending the data to a template.

The Model keeps all of the models in a file labeled “models.py”. Once a class is defined in this file, any objects created from each class will automatically be added to an SQLite database that is maintained inside the Django app. The requested data from the database will then be returned back to the View, and then returned to template.

Templates are used to dictate how the processed data will look on a webpage after it has been requested. A template consists of all of the basic utilities that can be included in any html document. Each page in a Django project will require its own template.

Urls.py

The url patterns in the “urls.py” file include “index”, “register”, “login”, and “create/personal-profile/”. Each url pattern sends a request to the View which calls a function by the same name in the “views.py” file. For example, the url pattern for “index” uses a line of code called “views.index”, and will call the function “index”, from the file “views.py”. This means that “views.py” will only consist of the 4 functions listed above. This “urls.py” file handles the url routing for the django application.

Models.py

In Django, database tables are created via python classes. These classes are individually referenced as a “model” and all together we call our database entries the “Models”. In the Alberta Science park IoT System we currently have many models of which let’s consider these three : Challenge Statement Data, Profile(User/Business) Data, and the User Data.

Views.py

“Views.py” will consist of 4 functions; “index”, “UserRegistrationView”, “UserLoginView”, and “PersonalProfileCreateView”. These functions return data to specific web pages within our web application. Each of the functions in the View are referenced by the “urls.py” file, and are called after being requested by the corresponding URL address in the Django application.

The View is not responsible for how the data is displayed, but rather what data will be displayed. The View is the section in which all of the functionality for the web application will reside. It is responsible for requesting queries from the database,

and then manipulating and organizing the data before passing it off to a React -Frontend for displaying to a browser.

Implementation Plan

The following section outlines the steps and milestones that need to be completed so that the Alberta Science Park System can be implemented on time. Figure 7.1 below depicts a rough project timeline and includes 10 major implementation milestones.

Conclusion

As a team, we feel that this project has great potential to very useful to our client. It will enable them to more effectively use their application to match-make possible personas and learn about the interests, and thus making a large impact in their productivity.

1. Compatibility Testing:-

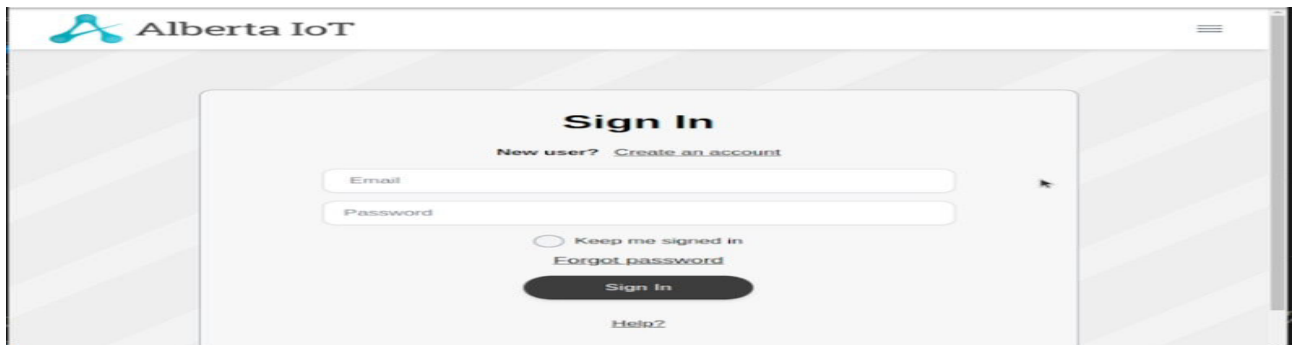
In this, we have to test the compatibility that the website is opening in the browser or not. So have tested the compatibility in the 3 of the top browsers named Google Chrome, Mozilla Firefox and Microsoft Edge. Name of the video is **Compatibility_Testing.mkv**

Google Chrome:

Have tested the compatibility in Google Chrome.

Expedted Result :- This Link should Get opened in the Google Chrome Browser.

Actual Result :- This link is opening in the Google Chrome Browser.



Mozilla Firefox:

Have tested the compatibility in Mozilla Firefox.

Expedted Result :- This Link should Get opened in the Mozilla Firefox Browser.

Actual Result :- This Link is opening in Mozilla Firefox Browser.

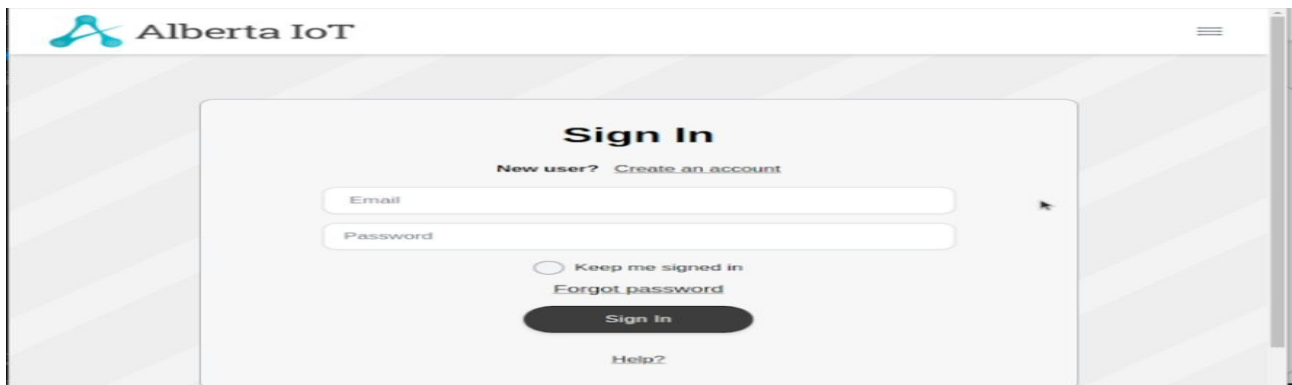


Microsoft Edge:

Have tested the compatibility in Microsoft Edge.

Expedted Result :- This Link should Get opened in the Microsoft Edge Browser.

Actual Result :- This Link is opening in Microsoft Edge Browser.



2. Penetration Testing:-

In penetration testing, I have scanned the ports in the website from 1 to 50000. In this no any port is open. Name of the video is **penetration_testing_port_scanner.mkv**

Have scanned to check that which ports are open in the website. No ports are open.

Expected Result :- No Ports should be opened.

Actual Result :- No Ports are opened

```
penetration_test.py > portscan
1 import socket
2 import time
3 import threading
4
5 from queue import Queue
6 socket.setdefaulttimeout(0.25)
7 print_lock = threading.Lock()
8
9 t_IP = "https://dev.thesciencepark.dev/"
10 #t_IP = socket.gethostbyaddr(target)
11 print ('Starting scan on host: ', t_IP)
12
13 def portscan(port):
14     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     try:
16         con = s.connect((t_IP, port))
17         with print_lock:
18             print(port, 'is open')
19         con.close()
20     except:
21         pass
22
23 def threader():
24     while True:
25         worker = q.get()
26         portscan(worker)
27         q.task_done()
28
29
30 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
c:/vscode/extensions/ms-python.python-2022.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 45815 -- /home/tecblcic/
Desktop/Projects/Alberta/penetration_test.py
Starting scan on host: https://dev.thesciencepark.dev/
Time taken: 2.44888247924892
tecblcic@tecblcic-HP-PC: ~/Desktop/Projects/Alberta : cd /home/tecblcic/Desktop/Projects/Alberta ; /usr/bin/env /bin/python3 /home/tecblcic
c:/vscode/extensions/ms-python.python-2022.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 41597 -- /home/tecblcic/
Desktop/Projects/Alberta/penetration_test.py
Starting scan on host: https://dev.thesciencepark.dev/
Time taken: 2.4548444747924892
```

3. GUI Testing:-

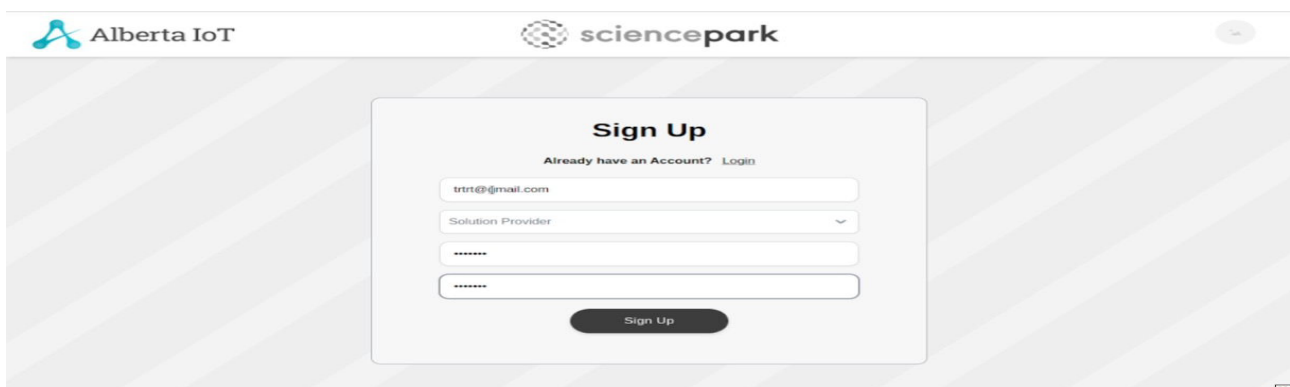
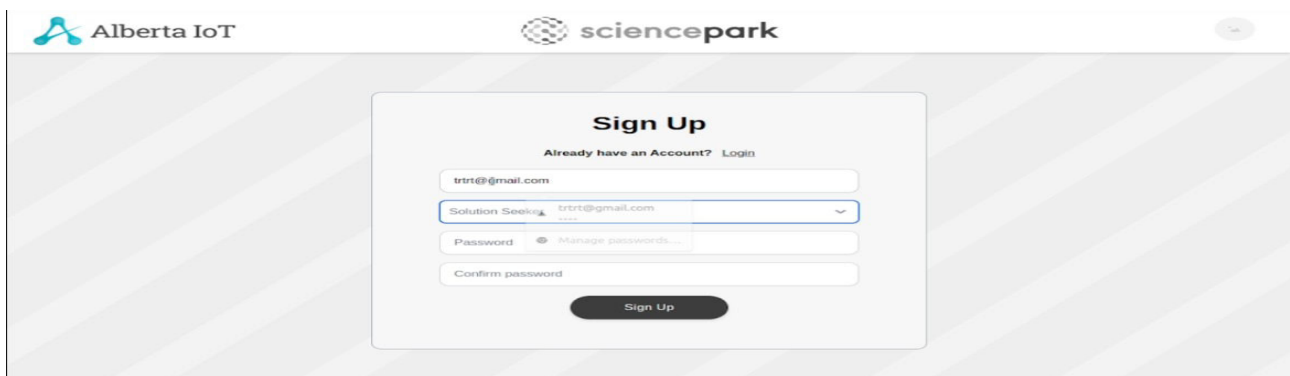
In this testing, we test the UI of the website. It generally evaluates the textboxes, buttons, links, icons, lists, etc. In this, I have tested Signup Page, Login Page, Create Profile Page, Edit Profile Page, Create Challenge and Business Profile Page. Names of the videos are **signup_page.mkv**, **Login.mkv**, **Profile_Create.mkv**.

Signup Page

Have tested the signup page testboxes and by adding the values.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.



Login Page

Have tested the Login page testboxes and by adding the values.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.

Alberta IoT sciencepark

Sign In

New user? [Create an account](#)

trtrt8090@gmail.com

Password

☐ Keep me signed in

[Forgot password](#)

[Sign In](#)

[Help?](#)

Alberta IoT sciencepark

Signed in successfully

Sign In

New user? [Create an account](#)

trtrt8090@gmail.com

☐ Keep me signed in

[Forgot password](#)

[Sign In](#)

[Help?](#)

Alberta IoT sciencepark

Signed in successfully

Profile Create Page:

Have tested the Profile Create page testboxes and by adding the values.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.



This screenshot shows the 'Profile Create' page for 'Alberta IoT' and 'sciencepark'. The page features a header with the logos and a user profile icon. The main form contains five input fields: 'First Name' (containing 'abcdefg'), 'Last Name' (containing 'pqrst'), 'Office phone Number' (containing '987'), 'Education' (containing 'Education'), and 'Position' (containing 'Position'). A 'Submit' button is located at the bottom of the form. A small '00/23' indicator is visible in the bottom right corner.



This screenshot shows the 'Profile Create' page with the same form as the previous one, but with different values entered: 'First Name' is 'abcdefg', 'Last Name' is 'pqrst', 'Office phone Number' is '987', 'Education' is 'Engineer', and 'Position' is 'abcd'. The 'Submit' button remains at the bottom.

Manual Testing (Alberta Science Park)

Testing Of Urls — To check whether each URL is correctly mapped to its corresponding view. We discover that all urls function correctly. And following the testing, I used Django coverage to assess the overall accuracy for the urls.py file. Django Coverage is a library that gives us the overall percentage of testing accuracy. Additionally, the overall accuracy is 100%.

Coverage report: 76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530

Module	statements	missing	excluded	coverage
ASP_SIT/ __init__.py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/ __init__.py	0	0	0	100%
api/admin.py	22	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	24	9	0	62%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_alter_personalinformation_first_name_and_more.py	4	0	0	100%
api/migrations/ __init__.py	0	0	0	100%
api/models.py	76	7	0	91%
api/renderers.py	9	0	0	100%
api/serializers.py	92	46	0	50%
api/tests.py	81	0	0	100%
api/urls.py	4	0	0	100%
api/utls.py	7	2	0	71%
api/views.py	129	56	0	57%
manage.py	14	2	0	86%
Total	511	122	0	76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530

Testing of Views – To confirm that each view is functioning properly. Every view stores data in the database and retrieves data from the database. The coverage report for the views.py file is 77%.

User Registration View:-

Expected Output

Current Output

HTTP_201_CREATED

HTTP_201_CREATED

User Login View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

User Password Reset View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

Coverage report: 76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530

Module	statements	missing	excluded	coverage
ASP_SIT/_init_.py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/_init_.py	0	0	0	100%
api/admin.py	22	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	24	9	0	62%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_alter_personalinformation_first_name_and_more.py	4	0	0	100%
api/migrations/_init_.py	0	0	0	100%
api/models.py	76	7	0	91%
api/renderers.py	9	0	0	100%
api/serializers.py	92	46	0	50%
api/tests.py	81	0	0	100%
api/urls.py	4	0	0	100%
api/utils.py	7	2	0	71%
api/views.py	129	56	0	77%
manage.py	14	2	0	86%
Total	511	122	0	76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530

Testing of Model – To verify that all models for building databases and tables are functioning properly.coverage report for models.py file is 91%.

Coverage report: 76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530



Module	statements	missing	excluded	coverage
ASP_SIT/__init__.py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/__init__.py	0	0	0	100%
api/admin.py	22	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	24	9	0	62%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_alter_personalinformation_first_name_and_more.py	4	0	0	100%
api/migrations/__init__.py	0	0	0	100%
api/models.py	76	7	0	91%
api/renderers.py	9	0	0	100%
api/serializers.py	92	46	0	50%
api/tests.py	81	0	0	100%
api/urls.py	4	0	0	100%
api/utils.py	7	2	0	71%
api/views.py	129	56	0	57%
manage.py	14	2	0	86%
Total	511	122	0	76%

coverage.py v6.4.2, created at 2022-07-18 19:29 +0530

Overall Report For Sprint 1

Overall coverage report for sprint 1 is 77%.

URL'S –

1

Test case is for the user registration url. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'register/'	For user registration	Passed

2

Test case is for the user login url. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'login/'	For user login	Passed

3

Test case is for the user password reset. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'reset-password/<uid>/<token>'	For user password reset	

VIEWS –

1

Test case for the user User Registration. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
UserRegistrationView	View for Register User	Passed	HTTP_201_CREATED	HTTP_201_CREATED

2

Test case is for the user User Login. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
UserLoginView	View for User Login	Passed	HTTP_200_OK	HTTP_200_OK

3

Test case is for the user User Password Reset. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
UserPasswordResetView	View for Password Reset	Passed	HTTP_200_OK	HTTP_200_OK

MODELS –

1

Test case is for the user User Creation. Test case for this MODEL passed.
Output matches the anticipated outcome.

Model Name	Description	Status	Expected Result	Actual Result
User	Creating User	Passed	HTTP_201_CREATED	HTTP_201_CREATED

2

Test case isfor the user Password Reset. Test case for this MODEL passed.
Output matches the anticipated outcome.

Model Name	Description	Status	Expected Result	Actual Result
User	Password Reset	Passed	HTTP_200_OK	HTTP_200_OK

1. Compatibility Testing:-

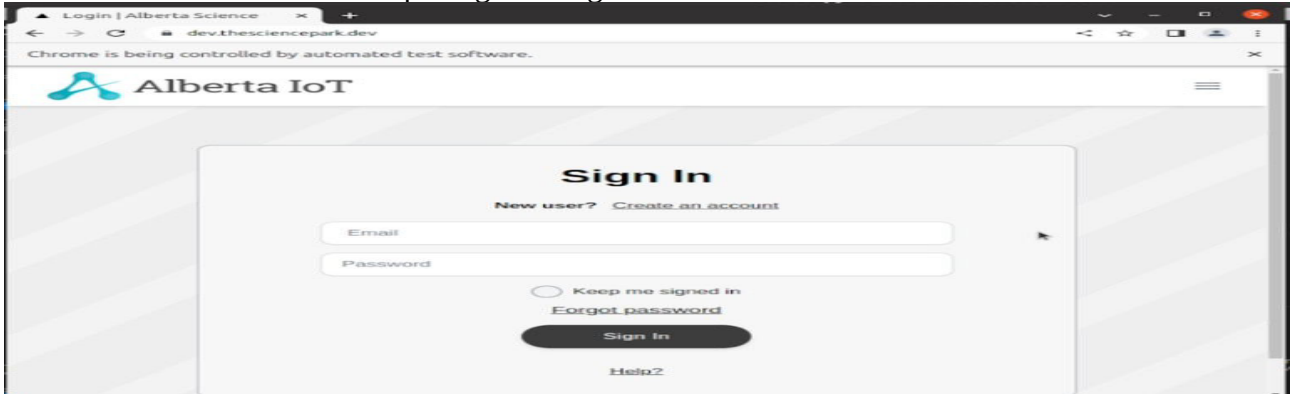
In this, we have to test the compatibility that the website is opening in the browser or not. So we have tested the compatibility in the 3 of the top browsers named Google Chrome, Mozilla Firefox and Microsoft Edge. Name of the video is **Compatibility_Testing.mkv**

Google Chrome:

Again after updation I've tested the compatibility of the website in this sprint in Google Chrome.

Expedted Result :- This Link should Get opened in the Google Chrome Browser.

Actual Result :- This Link is opening in Google Chrome Browser.



Mozilla Firefox:

Again after updation I've tested the compatibility of the website in this sprint in Mozilla Firefox

Expedted Result :- This Link should Get opened in the Mozilla Firefox Browser.

Actual Result :- This Link is opening in Mozilla Firefox Browser.

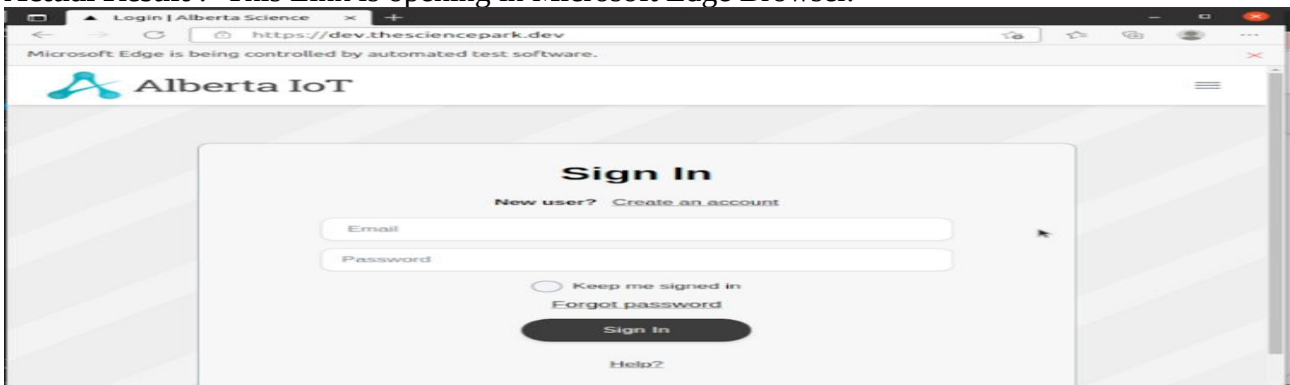


Microsoft Edge:

Again after updation I've tested the compatibility of the website in this sprint in Microsoft Edge.

Expedted Result :- This Link should Get opened in the Microsoft Edge Browser.

Actual Result :- This Link is opening in Microsoft Edge Browser.



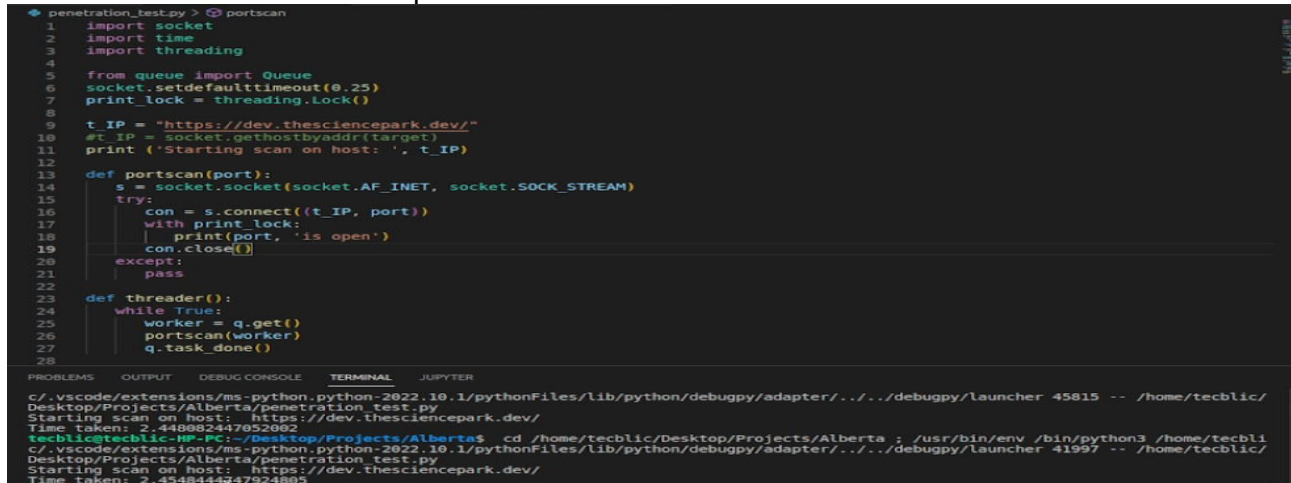
2. Penetration Testing:-

In penetration testing, I have scanned the ports in the website from 1 to 50000. In this no any port is open. Name of the video is **penetration_testing_port_scanner.mkv**

Again after updation of the website, I've tested the ports but no open ports were detected.

Expected Result :- No Ports should be opened.

Actual Result :- No Ports are opened



```
penetration_test.py > portscan
1 import socket
2 import time
3 import threading
4
5 from queue import Queue
6 socket.setdefaulttimeout(0.25)
7 print_lock = threading.Lock()
8
9 t_IP = "https://dev.thesciencepark.dev/"
10 #t_IP = socket.gethostbyaddr(target)
11 print ('Starting scan on host: ', t_IP)
12
13 def portscan(port):
14     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     try:
16         con = s.connect((t_IP, port))
17         with print_lock:
18             print(port, 'is open')
19         con.close()
20     except:
21         pass
22
23 def threader():
24     while True:
25         worker = q.get()
26         portscan(worker)
27         q.task_done()
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

c:/vscode/extensions/ms-python.python-2022.10.1/pythonFiles/lib/python/debugpy/adapters/.../debugpy/launcher 45815 -- /home/tecblc/Desktop/Projects/Alberta/penetration_test.py
Starting scan on host: https://dev.thesciencepark.dev/
Time taken: 2.448862447952092
tecblc@tecblc-HP-PC:~/Desktop/Projects/Alberta\$ cd /home/tecblc/Desktop/Projects/Alberta ; /usr/bin/env /bin/python3 /home/tecblc/Desktop/Projects/Alberta/penetration_test.py
Starting scan on host: https://dev.thesciencepark.dev/
Time taken: 2.4548444747924895

3. GUI Testing:-

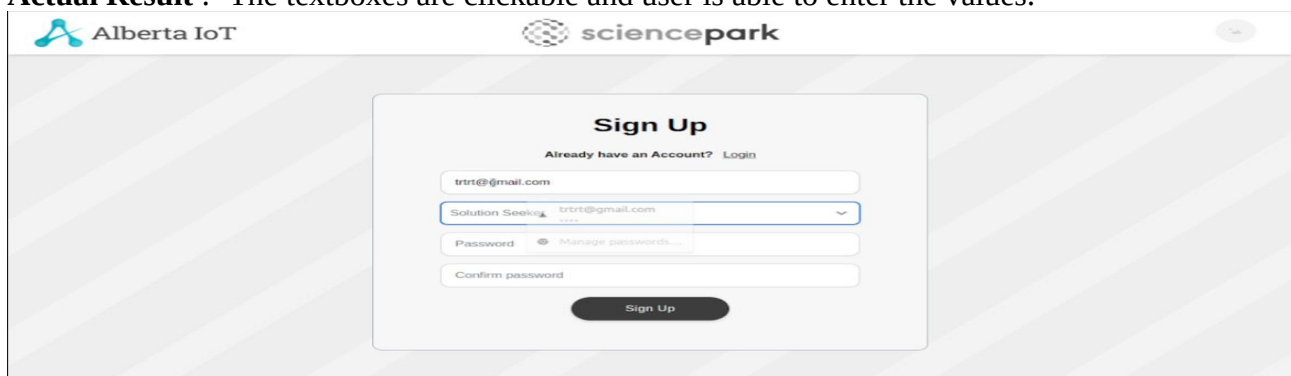
In this testing, we test the UI of the website. It generally evaluates the textboxes, buttons, links, icons, lists, etc. In this, I have tested Signup Page, Login Page, Create Profile Page, Edit Profile Page, Create Challenge and Business Profile Page. Names of the videos are **signup_page.mkv**, **Login.mkv**, **Profile_Create.mkv**, **Edit_Profile.mkv**, **Business_Profile.mkv**, **Create_Challenge.mkv**.

Signup Page

Again after updation I've tested the Signup Page of the website in this sprint.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.



The screenshot shows the 'Sign Up' page of the sciencepark website. The header includes the 'Alberta IoT' logo on the left and the 'sciencepark' logo on the right. The main content area has a light gray background with diagonal stripes. A white box in the center contains the 'Sign Up' form. The form has a title 'Sign Up', a link 'Already have an Account? Login', and four input fields: an email field with 'trtrt@gmail.com', a 'Solution Provider' dropdown menu, and two password fields with masked characters '*****'. A dark gray 'Sign Up' button is at the bottom of the form.

Login Page

Again after updation I've tested the Login Page of the website in this sprint.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.

The screenshot shows the 'Sign In' page of the sciencepark website. The header includes the 'Alberta IoT' logo on the left and the 'sciencepark' logo on the right. The main content area has a light gray background with diagonal stripes. A white box in the center contains the 'Sign In' form. The form has a title 'Sign In', a link 'New user? Create an account', and two input fields: an email field with 'trtrt80@gmail.com' and a 'Password' field. Below the password field are two links: 'Keep me signed in' (with a radio button) and 'Forgot password'. A dark gray 'Sign In' button is at the bottom of the form, and a 'Help?' link is below it.

The screenshot shows the 'Sign In' page of the sciencepark website after a successful login. The header includes the 'Alberta IoT' logo on the left and the 'sciencepark' logo on the right. A green notification banner in the top right corner says 'Signed in successfully'. The main content area has a light gray background with diagonal stripes. A white box in the center contains the 'Sign In' form. The form has a title 'Sign In', a link 'New user? Create an account', and two input fields: an email field with 'trtrt80@gmail.com' and a password field with masked characters '****'. Below the password field are two links: 'Keep me signed in' (with a radio button) and 'Forgot password'. A dark gray 'Sign In' button is at the bottom of the form, and a 'Help?' link is below it.



Profile Create Page:

Again after updation I've tested the Profile Create Page of the website in this sprint.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.

This screenshot displays the 'Profile Create Page' form. The form is set against a background of light gray diagonal stripes. It includes the following elements:

- First Name:** A text input field containing 'abcdefg'.
- Last Name:** A text input field containing 'qrst'.
- Office phone Number:** A text input field containing '987'.
- Education:** A text input field containing 'Education'.
- Position:** A text input field containing 'Position'.
- Submit:** A dark gray button with the text 'Submit'.

This screenshot shows the 'Profile Create Page' form after data entry. The form layout is identical to the previous one, but with the following updated values:

- First Name:** 'abcdefg' (unchanged).
- Last Name:** 'qrst' (unchanged).
- Office phone Number:** '987' (unchanged).
- Education:** 'Engineer'.
- Position:** 'abcd'.
- Submit:** A dark gray button with the text 'Submit'.

Edit Profile:

Have tested the Edit Profile page testboxes and by adding the values.

Expected Result :- The textboxes should be clickable and user should be able to enter the values.

Actual Result :- The textboxes are clickable and user is able to enter the values.

Alberta IoT sciencepark

abcdefg 987
Solution Provider

[Edit profile](#)

Email: trts8080@gmail.com Office phone Number: pqrs

Education: Engineer Position: abcd

Address

Languages

Alberta IoT sciencepark

First Name: abcdefgabcdefg Last Name: abcd Office phone Number: 995 Education: engineer

Position: abc Language Spoken: English Experience level: Junior Address Line 1: abcde

Address line 2: f City: Enter your city State: Enter your state Country: Enter your country

Submit

Dashboard Projects Challenges Profile Business Profile

Profile Updated Successfully

Loading...

Business Profile Page

Have tested the Business Profile page testboxes, Dropdown and by adding the values.

Expected Result :- The textboxes and Dropdown should be clickable and user should be able to enter the values in the textboxes.

Actual Result :- The textboxes and Dropdown are clickable and user is able to enter the values in the textboxes.

The screenshot shows the 'Business Profile' page of the 'sciencepark' application. The left sidebar contains a menu with 'Dashboard', 'Projects', 'Challenges', 'Profile', and 'Business Profile' (highlighted). The main content area contains a form with the following fields: Company Name (filled with 'ABCDEFGerr'), Company Email (filled with 'abcgrfgdefg@'), Company Phone (empty), Company Address 1 (empty), Company Address 2 (empty), Company Classification (dropdown menu with 'Select role' selected), Company City (empty), Company State (empty), Company Country (empty), and Company Description (empty). A 'Submit' button is located at the bottom right of the form.

The screenshot shows the 'Business Profile' page with test data entered into the form fields. The fields are: Company Name (filled with 'ABCDEFGerr'), Company Email (filled with 'abcgrfgdefg@getnadtga.com'), Company Phone (filled with '96545'), Company Address 1 (filled with 'abcdpfghqr'), Company Address 2 (filled with 'pgrxgfbyz'), Company Classification (dropdown menu with 'Construction' selected), Company City (filled with 'dehgfghi'), Company State (filled with 'jkghlm'), Company Country (filled with 'mnbhgd'), and Company Description (empty). A 'Submit' button is located at the bottom right of the form.

The screenshot shows the 'Business Profile' page after successful submission. The form fields are: Company Phone (empty), Company Address 1 (empty), Company Address 2 (empty), Company Classification (dropdown menu with 'Select role' selected), Company City (empty), Company State (empty), Company Country (empty), and Company Description (empty). A 'Submit' button is located at the bottom right of the form. A green circular icon and the text 'Profile Created successfully' are displayed in the top right corner of the page.

Create Challenge Page:

Have tested the Create Challenge page testboxes, Dropdown, Checkboxes and by adding the values.

Expected Result :- The textboxes, Dropdown and Checkboxes should be clickable and user should be able to enter the values in the textboxes.

Actual Result :- The textboxes, Dropdown and Checkboxes are clickable and user is able to enter the values in the textboxes.

Challenge Title
challenge123

Nature of Challenge/ Challenge Statement
abcdevj

Challenge Location
Select location

Industry
☐ Agriculture
☐ Oil and Natural Gas
☐ Manufacturing
☐ Retail
☐ Healthcare
☐ Real Estate
☐ Banking & Financial Services
☐ Insurance

Submit new Challenge

Up Coming Challenges

CHALLENGES	POST DATE
Dev Launch	July 2022
Alpha Launch	August 2022

Team Users

Team Users	Role
Admin	Manager
Admin	Manager

User management >

Challenge Title
Enter challenge

Nature of Challenge/ Challenge Statement
Write your statement here...

Challenge Location
Select location

Industry
☒ Agriculture
☒ Oil and Natural Gas
☒ Manufacturing
☒ Retail
☐ Healthcare
☐ Real Estate
☐ Banking & Financial Services
☐ Insurance

Submit new Challenge

Up Coming Challenges

CHALLENGES	POST DATE
Dev Launch	July 2022
Alpha Launch	August 2022

Team Users

Team Users	Role
Admin	Manager
Admin	Manager

User management >

Challenge Created successfully

00/30

Manual Testing (Alberta Science Park)

Testing Of Urls — To check whether each URL is correctly mapped to its corresponding view. We discover that all urls function correctly. And following the testing, I used Django coverage to assess the overall accuracy for the urls.py file. Django Coverage is a library that gives us the overall percentage of testing accuracy. Additionally, the overall accuracy is 100%.

Coverage report: 86%

filter...

coverage.py v6.4.2, created at 2022-07-29 12:50 +0530

Module	statements	missing	excluded	coverage
ASP_SIT/__init__.py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/__init__.py	0	0	0	100%
api/admin.py	27	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	20	6	0	70%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_rename_role_userrole.py	4	0	0	100%
api/migrations/0003_alter_user_role.py	4	0	0	100%
api/migrations/0004_user_is_admin_user_is_challenge_creator_and_more.py	4	0	0	100%
api/migrations/0005_rename_is_solution_seeker_user_is_solution_provider.py	4	0	0	100%
api/migrations/__init__.py	0	0	0	100%
api/models.py	96	6	0	94%
api/renderers.py	9	0	0	100%
api/serializers.py	211	41	0	81%
api/tests.py	115	0	0	100%
api/urls.py	4	0	0	100%
api/utils.py	7	0	0	100%
api/views.py	229	49	0	79%
challenge_creator/__init__.py	0	0	0	100%
challenge_creator/admin.py	4	0	0	100%
challenge_creator/apps.py	4	0	0	100%
challenge_creator/migrations/0001_initial.py	7	0	0	100%
challenge_creator/migrations/0002_industry_remove_challengestatement_description_and_more.py	5	0	0	100%
challenge_creator/migrations/__init__.py	0	0	0	100%
challenge_creator/models.py	26	2	0	92%
challenge_creator/permissions.py	31	18	0	42%

Testing of Views – To confirm that each view is functioning properly. Every view stores data in the database and retrieves data from the database. The coverage report for the views.py file is 79%.

Personal Profile Create View

Expected Output
HTTP_201_CREATED

Current Output
HTTP_201_CREATED

Profile Update View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

User Change Password View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

Business Profile Create View:-

Expected Output
HTTP_201_CREATED

Current Output
HTTP_201_CREATED

Business Profile Update View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

Send Password Reset Email View:-

Expected Output
HTTP_200_OK

Current Output
HTTP_200_OK

Coverage report: 86%

coverage.py v6.4.2, created at 2022-07-29 12:50 +0530



Module	statements	missing	excluded	coverage
ASP_SIT/ __init__ .py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/ __init__ .py	0	0	0	100%
api/admin.py	27	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	20	6	0	70%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_rename_role_userrole.py	4	0	0	100%
api/migrations/0003_alter_user_role.py	4	0	0	100%
api/migrations/0004_user_is_admin_user_is_challenge_creator_and_more.py	4	0	0	100%
api/migrations/0005_rename_is_solution_seeker_user_is_solution_provider.py	4	0	0	100%
api/migrations/ __init__ .py	0	0	0	100%
api/models.py	96	6	0	94%
api/renderers.py	9	0	0	100%
api/serializers.py	211	41	0	81%
api/tests.py	115	0	0	100%
api/urls.py	4	0	0	100%
api/utlis.py	7	0	0	100%
api/views.py	229	49	0	79%
challenge_creator/ __init__ .py	0	0	0	100%
challenge_creator/admin.py	4	0	0	100%
challenge_creator/apps.py	4	0	0	100%
challenge_creator/migrations/0001_initial.py	7	0	0	100%
challenge_creator/migrations/0002_industry_remove_challengestatement_description_and_more.py	5	0	0	100%
challenge_creator/migrations/ __init__ .py	0	0	0	100%
challenge_creator/models.py	26	2	0	92%
challenge_creator/permissions.py	31	18	0	42%

Testing of Model – To verify that all models for building databases and tables are functioning properly.coverage report for models.py file is 94%

Coverage report: 86%

 filter...

coverage.py v6.4.2, created at 2022-07-29 12:50 +0530

Module	statements	missing	excluded	coverage
ASP_SIT/__init__.py	0	0	0	100%
ASP_SIT/settings.py	33	0	0	100%
ASP_SIT/urls.py	3	0	0	100%
api/__init__.py	0	0	0	100%
api/admin.py	27	0	0	100%
api/apps.py	4	0	0	100%
api/managers.py	20	6	0	70%
api/migrations/0001_initial.py	9	0	0	100%
api/migrations/0002_rename_role_userrole.py	4	0	0	100%
api/migrations/0003_alter_user_role.py	4	0	0	100%
api/migrations/0004_user_is_admin_user_is_challenge_creator_and_more.py	4	0	0	100%
api/migrations/0005_rename_is_solution_seeker_user_is_solution_provider.py	4	0	0	100%
api/migrations/__init__.py	0	0	0	100%
api/models.py	96	6	0	94%
api/renderers.py	9	0	0	100%
api/serializers.py	211	41	0	81%
api/tests.py	115	0	0	100%
api/urls.py	4	0	0	100%
api/utils.py	7	0	0	100%
api/views.py	229	49	0	79%
challenge_creator/__init__.py	0	0	0	100%
challenge_creator/admin.py	4	0	0	100%
challenge_creator/apps.py	4	0	0	100%
challenge_creator/migrations/0001_initial.py	7	0	0	100%
challenge_creator/migrations/0002_industry_remove_challengestatement_description_and_more.py	5	0	0	100%
challenge_creator/migrations/__init__.py	0	0	0	100%
challenge_creator/models.py	26	2	0	92%
challenge_creator/permissions.py	31	18	0	42%
challenge_creator/renderers.py	0	0	0	100%

Final Report For Sprint 2

Final coverage report for sprint 2 is 86%.

URL'S –

1

Test case is for the user password reset and send email. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'send-password-reset-email/'	For user password reset	Passed

2

Test case is for create personal profile. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'create/personal-profile/'	Create personal profile for user	Passed

3

Test case is for update personal profile. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'update/personal-profile/<int:id>'	Update personal profile for user	Passed

4

Test case is for create business profile. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'create/business-profile/'	Create business profile for user	Passed

5

Test case is for update business profile. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'update/business-profile/<int:id>'	Update business profile for user	Passed

6

Test case is for upgrade role. Test case for this URL passed. Output matches the anticipated outcome.

URL Name	Description	Status
'upgrade-role/<int:pk>'	Upgrade role for user	Passed

VIEW'S

1

Test case if for the send reset password email. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
SendPasswordResetEmailView	View for Password reset and sends email	Passed	HTTP_200_OK	HTTP_200_OK

2

Test case is for the create personal profile. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
PersonalProfileCreateView	View for Creating personal profile	Passed	HTTP_201_CREATED	HTTP_201_CREATED

3

Test case is for the update personal profile. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
PersonalProfileUpdateView	View for Updating personal profile	Passed	HTTP_200_OK	HTTP_200_OK

4

Test case is for create business profile. Test case for this VIEW passed.
Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
BusinessProfileCreateView	View for Creating Business profile	Passed	HTTP_201_CREATED	HTTP_201_CREATED

5

Test case is for update business profile. Test case for this VIEW passed.
Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
BusinessProfileUpdateView	View for Creating Business profile	Passed	HTTP_200_OK	HTTP_200_OK

6

Test case is for upgrade role. Test case for this VIEW passed. Output matches the anticipated outcome.

View Name	Description	Status	Expected Result	Actual Result
RoleRegisterView	View for upgrade role	Passed	HTTP_200_OK	HTTP_200_OK

MODELS –

1

Test case is for the User Creation model. Test case for this MODEL passed. Output matches the anticipated outcome.

Model Name	Description	Status	Expected Result	Actual Result
User	Creating User	Passed	HTTP_201_CREATED	HTTP_201_CREATED

2

Test case is for the PersonalInformation model. Test case for this MODEL passed. Output matches the anticipated outcome.

Model Name	Description	Status	Expected Result	Actual Result
PersonalInformation	Model for creating personal profile	Passed	HTTP_201_CREATED	HTTP_201_CREATED

3

Test case is for the BusinessInformation model. Test case for this MODEL passed. Output matches the anticipated outcome.

Model Name	Description	Status	Expected Result	Actual Result
BusinessInformation	Model for creating Business profile	Passed	HTTP_201_CREATED	HTTP_201_CREATED

API DOCUMENTATION ---> <https://documenter.getpostman.com/view/21689200/VUjPGQUg>