

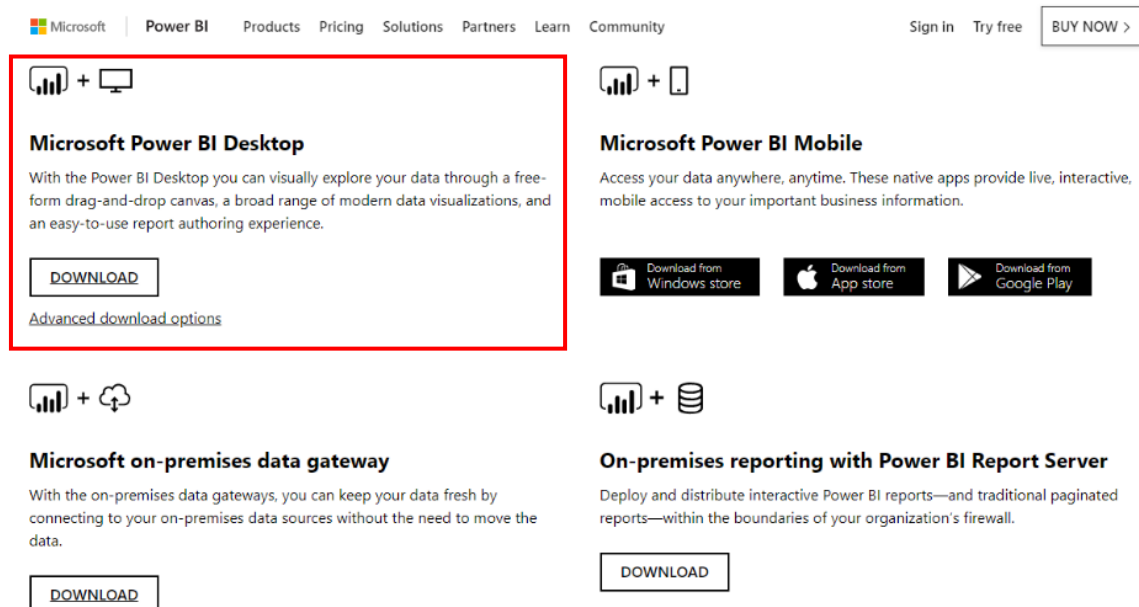
Patient Service Analysis Tool Deployment Manual

The following is a guide on every step needed to get the system up and running. This includes creating Power BI and Azure accounts, adding API keys to the code, setting up the database and installing required packages.

For information on how to refresh data, how to use Power BI, types of requests that can be made etc, please look at the User Manual. Before starting please be aware that in order to use Power BI, our visualisation tool of choice, you will need to be using windows, preferably windows 10.

Downloading and setting up Power BI

The first step needed is to download and install Power BI Desktop which is the free version of Power BI. You can use [this link](#) which will take you to a downloads page where you should then click on download Power BI desktop. This should then ask if you want to be redirected to the Microsoft store. Say yes and then download Power BI.



If you have a Power BI Pro subscription then by all means feel free to use it but it is not essential. If you don't already have a Power BI account you will need to create one. You will need to use a work or company email address to do this as you can't sign up using email addresses provided by consumer email services or telecommunication providers. This includes outlook.com, hotmail.com, gmail.com, and others. For detailed steps on how to create a Power BI account please see [this link](#). You now have Power BI downloaded and ready to use in later steps.

Creating an Azure account

The next step needed is to create an Azure account. Azure is used in this project to host a MySQL database, provide access to the Text Analytics API for sentiment analysis, and provide a storage container to upload data to be analysed.

Visit [this link](#) and click 'Start for free' to create your own Azure account.

Sign in with your Microsoft or GitHub account or create a free Microsoft account.

On the About you page, select your correct country or region. Enter your first and last name, email address, and phone number. Depending on your country, you might see additional fields, such as a VAT number. Select Next to continue.

On the Identity verification by phone screen, select your country code, and type the number of a telephone to which you have immediate access.

You have the option of text or call back to obtain a verification code. Select the relevant button, type the code in the Verification code box, and select Verify code.

If the verification code is correct, you're asked to enter details of a valid credit card. You will not be charged anything for signing up to Azure as Azure operates on a pay as you go basis. Azure also provides you with £150 of free credit when you sign up for the first 30 days. Enter the card information and select Next.

The last step is to review the agreement and privacy statement then select Sign up.

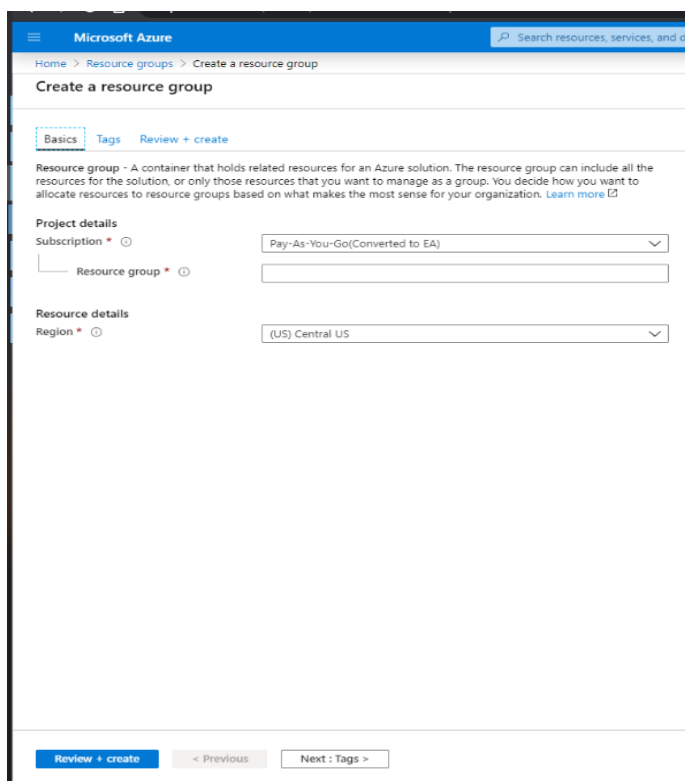
You have now successfully set up a free account on Azure and should be on the Azure portal home page.

Check for and create subscription if necessary.

On the Azure portal home click on 'Subscriptions'. Check to see if you have a subscription already, it may have been created when you set up your azure account. If the subscription is the trial account that lasts 30 days then that's fine as well. Before the trail is over Azure will prompt you to upgrade to a pay as you go subscription to continue using any services that use the subscription. If there is no subscription then click on 'Add' at the top left and follow the instructions to create an Azure subscription.

Create a Resource Group

Now you have an Azure account the next step is to set up a resource group. A resource group is a collection of resources that share the same lifecycle, permissions, and policies. To create a resource group first click on 'Resource groups' on the Azure portal home page. Then click on 'Create resource group'. Choose a subscription to use and enter the name you want to call your resource group into the field called 'Resource group'. Then for the field 'Resource details Region', select the appropriate region for you from the drop-down list. Then click 'Review and create', followed by 'Create'.



The screenshot shows the 'Create a resource group' page in the Microsoft Azure portal. The page has a blue header with the 'Microsoft Azure' logo and a search bar. Below the header, there's a breadcrumb trail: 'Home > Resource groups > Create a resource group'. The main heading is 'Create a resource group'. There are three tabs: 'Basics' (selected), 'Tags', and 'Review + create'. A descriptive paragraph explains that a resource group is a container for related resources. Below this, there are two sections: 'Project details' and 'Resource details'. In 'Project details', the 'Subscription' is set to 'Pay-As-You-Go(Converted to EA)' and the 'Resource group' name is empty. In 'Resource details', the 'Region' is set to '(US) Central US'. At the bottom, there are three buttons: 'Review + create' (blue), '< Previous' (disabled), and 'Next: Tags >' (disabled).

Create MySQL database on Azure

[This link](#) contains a detailed guide on how to set up a MySQL database on Azure. You should complete the section titled 'Create an Azure Database for MySQL server'. This will walk you through all the steps needed to set up the MySQL server.

When deciding on what pricing tier to choose [this link](#) provides information as to what each tier provides in terms of processing power and storage as well as price per month. Choose the plan that you think will be enough for your needs and the amount of data you plan on analysing and storing. For most cases the Basic tier should be sufficient.

You should then complete the section titled 'Configure a server-level firewall rule'. This creates a firewall rule for specific IP addresses so only registered IP addresses are able to make a connection to the database. Be sure to register all IP addresses that plan on using this database by following the instructions provided in that section.

You should then complete the section titled 'Connect to the server using the MySQL Workbench GUI tool'. This section walks you through how to connect to the MySQL server using MySQL Workbench. Once you have done this and have connected to the Azure MySQL server, click on File then New query tab. From here you should enter CREATE DATABASE <your database name> and execute the query by clicking on the lightning symbol. You should name the database yourself and will need to remember the name as it's needed later on.

The final thing you need to do is go back to Azure, click on your MySQL server resource that you just created, then in Overview change SSL enforce status to DISABLED.

You now have a MySQL server set up on Azure and have created a database on it.

Creating Azure blob storage container

[This link](#) contains instructions on how to set up a storage account on Azure but there are a few changes that need to be done. In the section 'Create a storage account', on step 7, for the field 'Account kind' don't leave it as 'StorageV2 (general-purpose v2)'. Instead change this to be BlobStorage. Step 8 can be skipped.

Other than the above changes, follow the steps in the link to create your storage account. Once you have done this go to the Overview page in your storage account resource. Click on 'Containers'. You then need to create a new container by clicking on the '+ Container' option. Name the container what you want but be sure to remember it as it will be needed later on.

In regard to how much the storage costs you can see [this link](#). It costs around £0.015 per GB of data uploaded.

You now have a storage account set up on Azure. The container you just made is where you will upload your Excel files containing patient feedback data to.

Creating Azure Text Analytics API resource

This section will provide instructions on how to create a resource for connecting to the Text Analytics API which is used to carry out sentiment analysis on the comments provided by patients.

From your Azure portal home page, go to add resource and search for Text Analytics. Click on Create and enter a name of your choice for the resource. Select a subscription to use and choose a location. For the pricing tier you can choose the free tier. This provides 5000 free transactions per month. If you do choose the free tier and exceed the amount you won't be charged. Instead, your limit will be reached, and you will no longer be able to make successful requests to the API. If you will be analysing more than 5000 new records of data a month then you should choose use the S-tier which is a pay as you go pricing tier. For more detailed

information on pricing for the Text Analytics API please see [this link](#). Choose a resource group to add this resource to and then click on Create.

Installing Python and required python packages.

If not already installed please visit [this link](#) to download python 3. You can check if python is installed by opening up command prompt and typing 'python --version'. If it gives you the version number of python then it is already installed, otherwise you will need to visit the previous link and install it yourself. You will also need to have PiP installed to download the python packages required. This should come by default with the python installation but can be checked similarly by running 'pip --version' on the command prompt. We have supplied a requirements.txt file with our code meaning that all you have to do to install the packages needed is to run 'pip install -r requirements.txt'. Before doing this though it may be best to set up a virtual environment so that the packages installed do not interfere with other versions of the same packages installed already.

To do this open a command prompt in the directory where the code is. In the command prompt type

```
pip install virtualenv
```

This installs the module that creates virtual environments. Once this is installed type

```
python -m venv venv
```

This creates a virtual environment called venv. To activate the virtual environment type

```
.\venv\Scripts\activate
```

This activates your virtual environment. If you install packages now they will be localised to the virtual environment and won't interfere with other projects outside of it. To install the required modules for the code run the following command.

```
pip install -r requirements.txt
```

This installs all the modules listed in the requirements.txt file which should be in the same directory as the code and the tests.

The next step is to fill in the template code with you Azure details.

Filling in required areas of the template code.

The code provided in the repository is currently configured for use on localhost. It can be changed in order for it to publicly hosted on a platform such as Heroku but for the purposes of this setup, I will go through how to get the system ready for running on localhost.

Once you have all the code from GitHub on your local machine there are multiple places which you need to fill in information regarding the resources you set up previously on Azure, so the system knows what database to use, where the data is stored etc.

AzureBlobStorage.py

In either a text editor such as Atom or an IDE such as PyCharm or Visual Studio, open up the project folder containing all the code. Then open the file named AzureBlobStorage.py. In this file you will need to make 3 changes to the __init__() constructor method. The variables storage_account_name, storage_account_key and container_name have default values 'your_storage_account_name', 'your_storage_account_key' and 'your_container_name' respectively.

These default values need to be replaced with the details for your Azure storage account resource. Your storage account name and container name are the names you gave the resource and the container respectively. When you are in the storage account resource, down the left under settings you will find an option called Access keys. Choose key1 as your storage account key. Make sure that the details for your accounts are still enclosed within quotes in the same way the default values were. Once the account name, account key and container name have been replaced with values for your specific Azure resource you are done with this file.

TextAnalytics.py

In this file there are 2 changes that need to be made, both are in the `__init__()` constructor. The variables `subscription_key` and `endpoint` have default values `your_subscription_key` and `your_endpoint`. When you are in the Text Analytics resource you made, down the left under Resource Management you will find an option called Keys and Endpoint. Choose Key1 to use as your subscription key and copy this and the endpoint into their respective places in the code template ensuring they are still enclosed in quotes. This API was chosen because after testing it out on 20,000 pieces of data, we found it was able to correctly determine the positivity of a patients comment 94.5% of the time.

Database.py

In this file there are 5 changes that need to be made with an additional optional change. In the method called `connect_to_database()`, on lines 10 to 13 you need to replace the default values for user, password, database and host with corresponding values for your database on Azure.

Your username and hostname (host) can be found by opening MySQL Workbench, right clicking the connection you made earlier and selecting edit connection. Once you have these details you can then click close and exit Workbench.

Alternatively, you can go to the MySQL Server resource you made earlier in Azure portal and in the Overview section you can find the value for host under 'Server name' and the value for user under 'Server admin login name'.

The value for password is the password you used when you created the resource and the value for database is the name you used earlier on when creating the database on MySQL Workbench.

The last necessary change that you need to make is in the `create_table()` method on line 57. The line has "USE 'your_database_name'" in it. You should replace `your_database_name` with the name of your database. As in previous sections, ensure that the replacement of the default values are all still within quotes.

An optional change that can be made is changing the name of the table the data is stored in. By default, it is called `feedbackdatabase`. You can use the find and replace tool in order to replace all instances of `feedbackdatabase` with whatever name you want to call your table in your database where the data will be stored in. This change isn't necessary though and the system will run fine without this change.

PatientServiceAnalysisTool.py

There are 2 changes to be made in this file. In the method `get_password()`, on lines 23 and 24 there are the default values `your_chosen_username` and `your_chosen_password`. You need to choose a username and password that will need to be supplied every time a request is made to the API to get data. This is to add a layer of security to prevent unauthorised access. This is in addition to the security measure where in order to access the database, the IP of the source of the request must be registered with the Azure MySQL server resource under Connection Security.

Replace these 2 fields with a username and password of your choice, again remembering to enclose them in quotes as they were before.

TestPatientServiceAnalysisTool.py

In the method `test_successful_authentication()`, on lines 22 and 23 are the same default values as before, `your_chosen_username` and `your_chosen_password`. Replace these with the username and password you just chose in the last section.

Request.py

In this file there is once again multiple places with the default values `your_chosen_username` and `your_chosen_password`. Go through and replace them all with the username and password you chose for the API. This file contains 4 examples of requests so there should be 4 places where you need to replace the default values with your values for the username and password.

Running tests and checking everything is working

The project folder includes a set of test files that can be run to check that everything is working as expected. These tests ensure that connections can be made to all the Azure resources, the keys, passwords and endpoints were entered correctly, and that authentication is set up for the API.

To run the tests, open up a command prompt in the same directory as where the code is. You should then type the following command into the terminal:

```
python -m unittest discover
```

This will then run all the unit tests they should hopefully all pass.

Common reasons for test failures

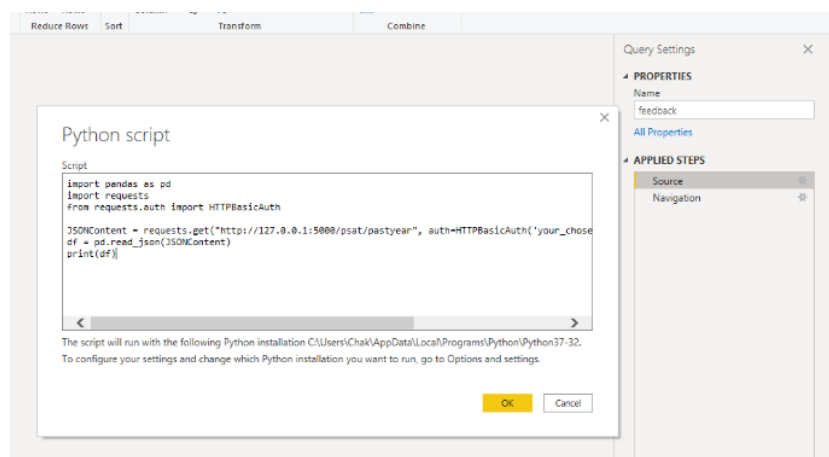
Some common reasons that tests may fail include:

- Incorrect details for API key or API endpoint, double check that the API keys and endpoints you are using in the code are the exact same as the ones shown on Azure. Also check your details for the database connection is correct and everything is spelt correctly.
- Not surrounding your Azure details such as keys and endpoints in quotes.
- Your IP address is not registered on Azure. Go to your MySQL Server resource and on the left under Settings you'll find Connection security. Click on this then click on '+ client IP'. This will automatically fill in your IP address so all you need to do is add it. Allow it to update then try to run the tests again.
- SSL enforce status set to ENABLED. Go to your MySQL server resource and in overview check to see what your SSL enforce status is. It should be DISABLED. If it's not, click on it to change it to DISABLED.
- Mismatching passwords and usernames. Make sure you have spelt your username and passwords correctly in all places where they are used. The username and password for the API should be the same as the one in `TestPatientServiceAnalysisTool.py`.
- Missing libraries. Make sure you have run `pip install -r requirements.txt` in the correct directory to install all the required packages.
- Running unittest command in wrong directory, make sure it's run in the same directory as the project code files.

Adding python request script to Power BI

Assuming the tests ran successfully, you have now finished setting up the code. The last step in the set up is to configure Power BI to make a request to the API in order to refresh its data. To do this open up the Power

BI file called PatientServiceAnalysisTool.pbix in Power BI. On the left find the Fields tab and right click on the table called 'feedback' then select 'edit query'.



In the dialogue box that pops up, double click on 'Source' and a box called 'Python script' should pop up. The box will have some python code in it already, you should get rid of this code as you will be replacing it later on. In this box you should add the code shown below. This code will be run whenever you refresh data on Power BI, manually or automatically. The script makes a request to the API on localhost for the most recent years

worth of data. If later on you publicly host the API somewhere else then be sure to change the host from local host to whatever the hostname of the platform you hosted the API on is. You can also use different endpoints to get different amounts of data to analyse, this is explained further in the User Manual.

Code to copy into 'Python Script' box:

Please copy everything between (but not including) the dashed lines into the 'Python script' box but don't click on ok yet. Also be sure to replace your_chosen_username and your_chosen_password with your username and password that you chose for your API in the section titled PatientServiceAnalysisTool.py

```
import pandas as pd

import requests

from requests.auth import HTTPBasicAuth

JSONContent = requests.get("http://127.0.0.1:5000/psat/pastyear",
auth=HTTPBasicAuth('your_chosen_username', 'your_chosen_password')).json()

df = pd.read_json(JSONContent)

print(df)
```

After you have copied the code, you need to run the API on localhost. To do this open up a command prompt in the project code directory and type:

```
python PatientServiceAnalysisTool.py
```

This should start up the API on your localhost on port 5000. You should now click on ok on Power BI at which point the script will run. At this point you should not have any data uploaded to your Azure storage resource so this should not take much time. Once it is done, press the red X on the top right, this will ask you if you want to apply the query changes, select yes.

From now on when you add data to your Azure storage account and want to update Power BI to see the visualisations, you must run the API on localhost if you are not publicly hosting it elsewhere, and then click Refresh at the top, under the home tab. This will re-run the script and process the data and visualise it. More information on this is included in the User Manual.

You have now set up all the resources necessary to use the system. For details on how the system works and how to use it please refer to the User Manual.