KINCYCL PEDIA

Documentation

Developed by: Abhilash R S, Arun R, Samia A, Solomon P, Tong M.

About

When the complete human genome was sequenced, research into identification and characterisation of human protein kinases became more prevalent. Kinases are a protein class that chemically add phosphoryl groups to themselves (auto-phosphorylation) or other proteins (substrates) through a process called phosphorylation. The sites where phosphorylation occurs are defined as phosphosites.

Kincyclopedia is a web application developed by a group of five students part of the MSc Bioinformatics course from Queen Mary University of London, under the supervision of Prof Conrad Bessant and Dr Fabrizio Smeraldi. The ultimate aim of our application is to act as a central hub for information on human kinases, all of their known phosphosites and kinase inhibitors.

An important feature apart from the search function is the phosphoproteomics data analysis tool, which produces a result in three categories bar plot representing the relative kinase activity and relative inhibited kinase activity, volcano plot for the complete protein and a volcano plot for all the identified kinases. All the files which were used to these plots can be downloaded for further use.

INDEX

List of Figures

1. Software Design:

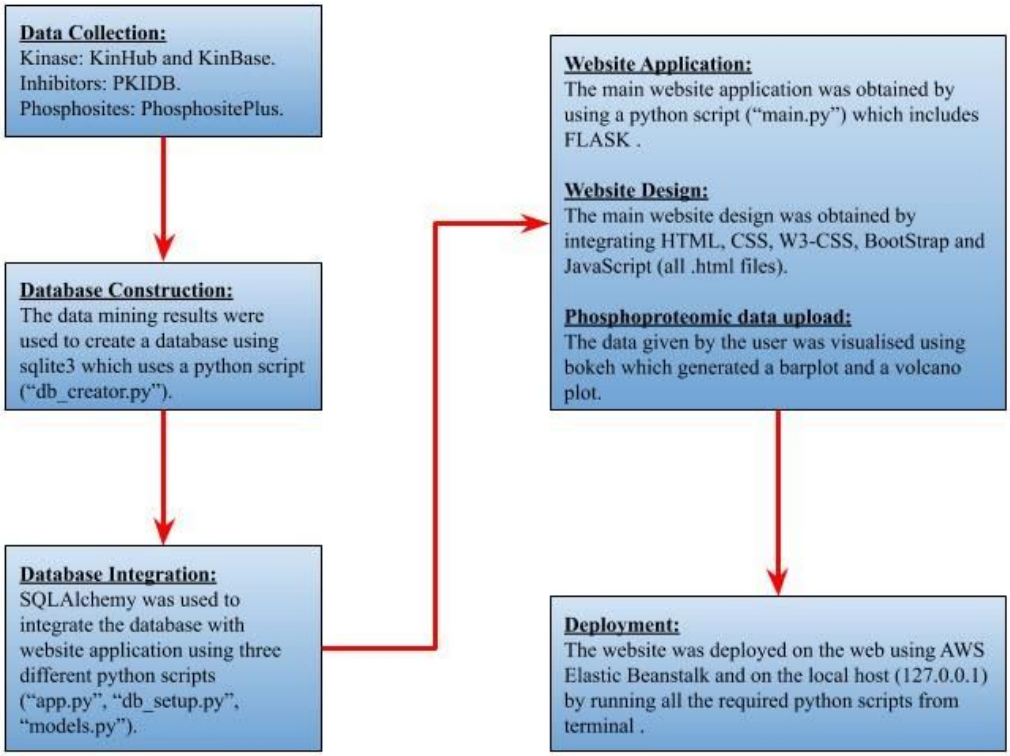

Figure 1: Diagram illustrating the software design and integration of the Kincylopedia web application. Red arrows indicate the directional workflow to reach the end product.

The initial steps in designing the website were the mining and cleaning of the data collected from KinHub (1), KinBase (2), UniProt (3), Protein Kinase Inhibitor Database (PKIDB) (4) and PhosphositePlus (5). The cleaned data was then used for constructing a database using SQLite3 (6). The database was integrated into the website using SQLAlchemy (7) and Flask-SQLAlchemy (8). We then used WTForms (9) to create the kinase, inhibitors and phosphosite search functions, which were able to acquire all the information derived from the integrated database. Finally, a functional web application was created using the Flask (10) package in python as a template engine. The design of the website and the website routes were obtained using HTML language (11), various static files like CSS (12), BootStrap (13), W3-CSS and JavaScript (14) files, which were linked to the main application. A combination of these files was able to generate an overall website which was deployed on the web using Amazon Web Services - Elastic Beanstalk (15). This application can also be deployed on localhost (i.e 127.0.0.1 or 0.0.0.0) by running all the python scripts necessary to run this application.

2. Website Architecture:

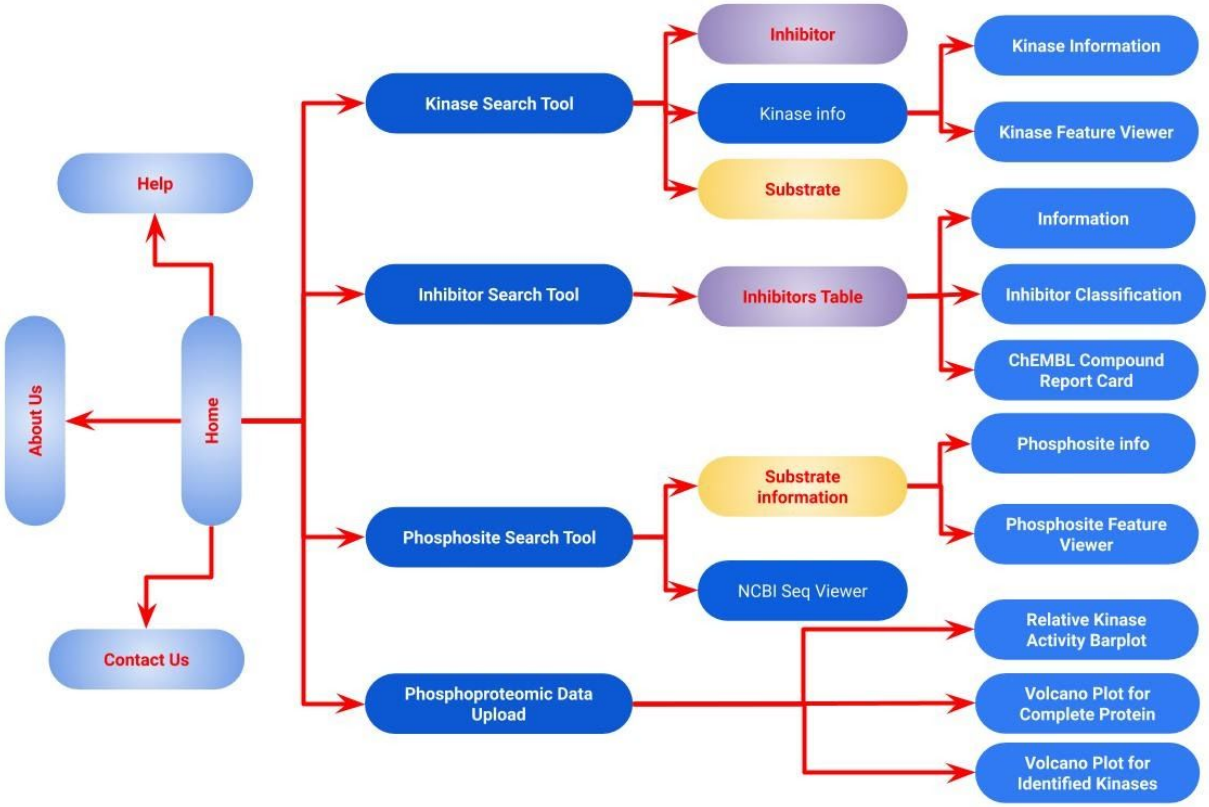

Figure 2: Diagram to illustrate the website architecture of Kincyclopedia.

The website architecture is as follows:

- The home page is linked to about us, contact us and help pages (in light blue). It also displays the Kinase search function, inhibitor search function, phosphosite search function and the phosphoproteomic data upload function.
- The kinase search results were divided into kinase table, inhibitor table (linked to the inhibitor search results, labelled in lilac) and phosphosite/substrate table (linked to substrate search results, labelled in yellow).

3. Data Collection:

3.1. **Kinase Data**

The bulk of the information for the initial 536 human kinases was collected from Kinhub (1) using the read_html function of pandas (16). We chose this source as it appeared to be one of the most recently updated platforms with the relevant information having last been updated/used in a publication by its creators in 2017 (17). Additionally, all of the listed kinases are already well-annotated on UniProt with each of them having their own UniProt IDs already assigned. In addition, to the IDs we also gathered; gene name, protein name and the classification group, family and subfamily for each of the kinases.

We then collected a list of aliases for each kinase from Kinbase (2) again using the read_html function of pandas and then merged it with our table of data from Kinhub. Further information such as the Uniprot entry names and the Subcellular Location for each kinase were obtained by running our UniProt IDs through the UniProt API (3).

Data was then curated, with 21 kinases being removed making our final list of human kinases 515. They were removed either due to them being pseudogenes of other kinases that were already listed in our table or because the kinase was not adequately annotated in UniProt.

3.2. **Phosphorylation Site Data**

The information for phosphorylation site data was downloaded from PhosphoSitePlus (5) with the entirety of the data being comprised from two files: Kinase_Substrate_Dataset.gz and Phosphorylation_site_dataset.gz, which were then converted to CSV format so that they could be cleaned using pandas.

In the Kinase_Substrate_Dataset, there were just under 18,500 rows of data. We noted that a significant proportion of them had kinase and substrate information from organisms that were not human. As we wanted our web application to be a source to access human kinase information, we decided to filter the table to contain only human kinases and substrates. This reduced the number of rows of data down to just under 11,000. From there we obtained the gene name and UniProt IDs of the kinases and substrates, the substrate protein name, the modified residue (with position number) and the neighbouring 7 amino acids either side of the phosphorylation site.

We then collected the chromosome location of our phosphosites from the Phosphorylation_site_dataset file, as well as substrate UniProt entry names using the UniProt

API, and merged it with our clean Kinase_Substrate_Dataset table to have our completed Phosphosite table.

3.3. **Kinase Inhibitor Data**

We collected information for human kinase inhibitors from the PKIDB (4), a resource that is curated and updated with the last update for the database being conducted relatively recently on 11th December 2019. We extracted this content of the database and put it into a CSV format so that we could clean it using pandas. The number of rows from this table at this point was 218.

For our database, we required each kinase target to be listed as an individual row entry in order for us to connect our Kinase and Inhibitor tables together. PKIDB presented this data in a list format so we used pandas to reformat the data in such a way that it was easy to construct our database. Due to this reformatting, the number of rows for our inhibitor table increased to 493 rows.

From PKIDB, we obtained information on the International Nonproprietary Name (INN) of the inhibitor, an image link to the chemical structure of the inhibitor and several pharmacological metrics that we assessed as being important information that a user interested in kinase inhibitors would like to view including Violations of Lipinski's Rule of Five (RoF), Molecular Weight (MW), Log of Partition Coefficient (LogP), Topological Polar Surface Area (TPSA), number of Hydrogen Bond Acceptors (HBA) and Hydrogen Bond Donors (HBD), Number of Rotatable Bonds (NRB), the Simplified Molecular Input Line Entry System (SMILES) and the IUPAC International Chemical Identifier (InCHI key).

Additionally, we used the regex, requests (18) and beautifulsoup4 (19) package to extract the ChEMBL IDs of those inhibitors in the PKIDB that had them. However, a few of the inhibitors in PKIDB did not have ChEMBL IDs. As we wanted to ensure inhibitors in our database were verified and annotated from multiple sources, we chose to remove these entries from our list, thereby reducing the number of rows down to 469, for our final inhibitor table.

4. Database Construction:

With SQLite3 (6) being a free open source database management system, and due to the size of the data populating our database being relatively small, we decided to use it to construct our database. With SQLite3 already being a part of the Python standard library and also being portable to SQLAlchemy (7), to make the future steps of integration with the front end of our web app relatively simple to carry out. Other advantages over other database management systems are that it does not require a server to run. It can also be run locally on your

laptop/desktop computer, and it does not require a user to set up an account to access the database.

With SQLite3, we first set up three empty tables to contain our respective Kinase, Phosphosite and Kinase Inhibitor data and then using the CSV module of the Python library (20), we read our cleaned CSV tables as a dictionary with our headings as the keys and the contents of the column as our values, using the DictReader function. Finally, we insert the dictionary values into each column of our empty database tables, in order to populate our three tables with data.

From here, all that was left to do was to connect our three tables together in order to create a functional database. We noted that the relationship between our Kinase to Inhibitors tables and our Kinase to Phosphosite tables can be considered to have a many to many relationships i.e. a kinase can have more than one inhibitor and an inhibitor can act on more than one kinase, in the case of the relationship between our Kinase to Inhibitors tables. As a result, we needed to create a relational table for each set of two tables so that querying data between the tables is carried out more efficiently. To populate the relational tables, we used the primary keys of each respective table as foreign keys and connected the tables by the data in common between each pair of tables. We then used the DB Browser for SQLite (21) to visually check the database to make sure the tables and connections were done correctly prior to integration. The full database and its connections are illustrated in our schema in Figure 3.

5. Database Integration:

SQLAlchemy (7) was used to create an engine to communicate with our SQLite3 database file, create a session to act as an intermediary where queries are passed on to the engine, and a base where our database tables could be directly imported to. We then used the Flask-SQLAlchemy (8) extension of Flask (10) to integrate all of our SQLAlchemy files to the front end of the website.

When querying our database through Flask-SQLAlchemy we used the .filter function to filter our database based on the user inputted search string. We also used the .join function when we needed to obtain query results from multiple tables from our database.

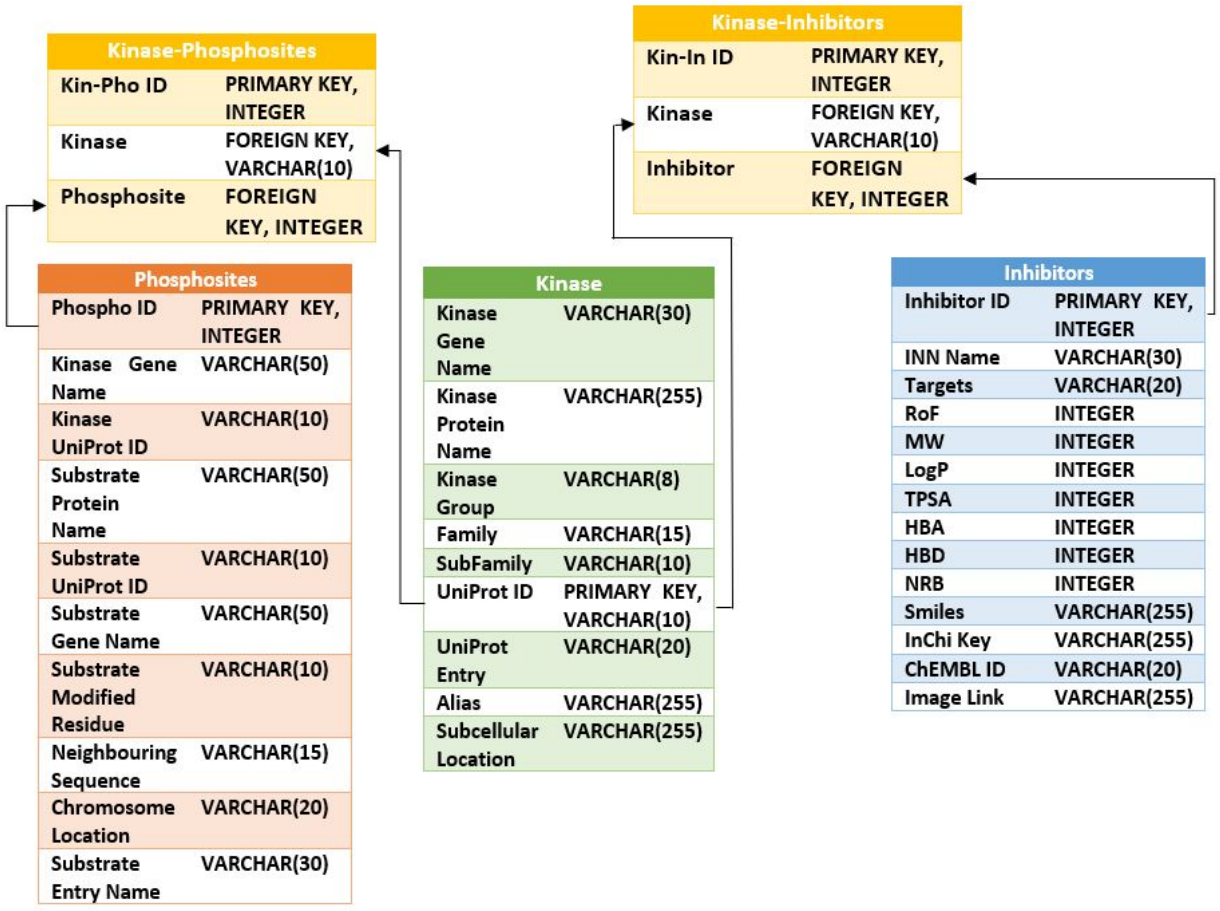

Figure 3: Database Schema

6. Web Application:

A responsive web application was created using Flask (10), which is integrated with all the HTML templates (11). This contains the layout and the information that it needs to display on the website. The main flask application script contains all the different routes, which are useful to hyperlink to different sections of the website. The main flask application script also contains different routes defined for all the simple queries, which takes the search strings entered into different search functions and creates different hyperlinks to its respective report cards. A similar process was followed for the data upload, where a route was defined for data upload that follows into the data results and the file uploaded is saved into the static folder. This is used for data visualisation.

The web application focuses on integrating three independent search functions that include a Kinase search, an Inhibitor search and a Phosphosite search as well as a section that allows the user to upload their own Phosphoproteomic data. Each search function further deviates into three

search options that include Kinase Gene Name, Kinase Uniprot ID and Alias for the Kinase search. INN, ChEMBL ID and cognate targets for the Inhibitor search, and Substrate Uniprot ID, Substrate Gene Name and Substrate Protein Name for the Phosphosite search.

6.1. **Search Functions** (Kinase, Inhibitor, Phosphosite)

For the **Kinase search function**, we included three different tables that contain information about kinases, inhibitors and phosphosites respectively:

- The kinase table includes the Kinase Name, UniProt ID, and Aliases, where kinase name was hyperlinked to its respective Kinase report cards.
- The inhibitor table includes Inhibitor ID (primary key generated during database construction), Inhibitor name, ChEMBL ID, and Targets, where Inhibitor ID was hyperlinked to its respective Inhibitor report cards.
- The phosphosite table includes Phospho ID (primary key generated during database construction), Kinase gene name, Substrate gene name, Substrate modified residue, Chromosome location where Phospho ID was hyperlinked to its respective Phosphosite report cards.

For the **Inhibitor search function**, we included an inhibitor table which contains Inhibitor ID, Inhibitor Name, ChEMBL ID, and Targets, where Inhibitor ID was hyperlinked to its respective Inhibitor report cards.

For the **Phosphosite search function**, we included a genome browser from NCBI Sequence Viewer (22) that allows the user to visualize molecular information, sequence data and annotation. A protein can be visualized in the sequence viewer by searching the protein in its respective chromosome, which can be obtained from the Phosphosite search results page.

6.2. **Report Cards** (Kinase, Inhibitor, Phosphosite)

The **Kinase report card** was divided into two components:

- Kinase information: This tab contains protein name, gene name, aliases, UniProt ID, UniProt entry name, classification of the kinase (group, family and sub-family), and subcellular location.
- Kinase feature viewer: This tab includes a feature viewer obtained by ProtVista (23) (A Bio Javascript), which uses the kinase UniProt ID to generate a UniProt feature viewer.

The **Inhibitor report card** was divided into three components:

- Inhibitor information: This tab contains the inhibitor name and a 2D image of the compound.
- Inhibitor classification: This tab contains SMILES, INCHI key (chemical notations of the inhibitors) and targets.
- ChEMBL compound report card (24): This tab includes a compound report card which was embedded into the website using the link from ChEMBL database.

The **Phosphosite report card** was divided into two components:

- Phosphosite information: This tab contains kinase and substrate gene name, kinase and substrate UniProt ID, kinase and substrate UniProt entry name, substrate name, substrate modified residue, chromosomal location and neighbouring sequence (includes +/- 7 amino acid residues).
- Phosphosite feature viewer: This tab includes a feature viewer obtained by ProtVista (A Bio Javascript) which uses the Phosphosite UniProt ID to generate a UniProt feature viewer.

6.3. ProtVista Feature Viewer

ProtVista Feature Viewer (23) was included because of interactive feature categories such as:

- Domains and sites: This feature provides the position and type of each domain (repeat, calcium-binding, DNA binding etc), the binding site for any chemical group (coenzyme, prosthetic group etc).
- Post-translational modifications: This feature provides modified residues (excluding lipids, glycans and protein cross-links), lipidation, glycosylation, disulfide bonds, cross-links.
    - Modified residues contain the site of phosphorylation, description as well as literary evidence, which supports the use of ProtVista.
- Structural features: This feature provides alpha-helix, loops and beta-sheet.

6.4. NCBI Sequence Viewer

NCBI sequence viewer (22) was included because of its ability to browse a particular substrate gene name using its genomic location (chromosomal location). It includes all the 22 chromosomes along with X, Y and MT chromosomes. This provides all the genes, biological regions that are associated with NCB*I Homo sapiens* annotation release 109.20191205. The substrate gene name search result provides the genomic information like gene position,

nucleotide length, links and tools (HGNC, OMIM, FASTA, GenBank, etc). The sequence provides an interactive mode which includes:

- Custom set of relevant annotations
- Zoom into the area of interest (chromosomes and locus)
- Setting genetic and mutational markers to illustrate the focus of results
- Interactive hover tool, which can show more detailed information and hyperlinks to relevant resources.

6.5. **Phosphoproteomic Data Upload and Analysis**

The phosphoproteomics data upload tool allows the user to upload a TSV file in order to visualise their data, identify the kinases acting on their substrates and subsequently find the relative kinase activity of each kinase in their sample. The phosphoproteomics data accepted by this tool has a control mean and inhibitor mean. As such, the data is expected to comprise of columns in the order of: Substrate, Control mean, Inhibitor mean, Fold change, p-value, Control coefficient of variance and Inhibitor coefficient of variance, where the latter two columns are optional. Files with multiple inhibitor data are also accepted, however only the first inhibitor will be analysed.

The dataset the user uploads is opened using the pandas python module as the data can be loaded into a data frame as a data frame allows for ease of data manipulation. Initially the data was filtered by filling in any blank entries in the coefficient of variance columns (if the file contains these) with a value of zero and then removing any rows in the dataframe with blank entries. This is to ensure that only the substrates with control mean, inhibitor mean, p-value and fold change entries remain in the dataset, as these are the data values needed for visualisation. After this, any entries where the substrate modified residue is ‘None’ are also removed, as a lack of modified residue means that the substrate’s corresponding kinase cannot be found. Substrates with a methionine residue are also removed from the dataset, using a regular expression, due to stable phosphorylation occurring at serine, threonine and tyrosine residues (25) while the methionine residues detected by MS-LC are a result of them being oxidised due to their close proximity to a phosphosite (26). The substrate column was then split using two regular expressions so that a new column for the modified residue (titled ‘Phosphosite’) was created and the substrate column was left with entries comprised of the substrate gene name or its Uniprot entry name. Finally, entries that had a Uniprot entry name were converted to their gene name using Uniprot API, where the entries were selected by a regular expression using their module and gene names retrieved with the requests module. Substrates, where the Uniprot entry names yielded no gene name, were removed from the dataframe as substrate gene name and modified residue are needed to find the kinase.

After the data is filtered, it is now in the format needed to identify the kinases. Using Kincyclopedia's database, a data frame was created using the sqlite3 and pandas modules where the kinase gene name, substrate gene name and substrate modified residue data was extracted from the PhosphoSites table. This dataframe was then merged with the user data to find the kinases by setting the substrate gene name and substrate modified residue as the dataframe index.

Relative kinase activity can be determined from phosphoproteomics data as each phosphorylation event is the result of the activity of a kinase (27). For our data analysis we decided to use the kinase activity analysis (KAA) method proposed by Qi et al (28) where the relative activity of a kinase can be determined by finding the number of links it has in a substrate-kinase network and comparing it to the background activity. To do this the sum of the control mean for each kinase was found by using the numpy package. This was divided by the sum of all the control means in the data frame produced after kinase identification. The same was done to the inhibitor data (using inhibitor mean and the sum of all inhibitor means) allowing for the data to then be plot on a graph. The data frame was then sorted by descending relative kinase activity.

Three graphs were chosen to represent the data: a bar graph and two volcano plots using the bokeh package (29) due to plots in bokeh having interactivity and having the ability to be embedded into a HTML. The bar graph shows the relative kinase activity and the relative inhibited kinase activity of the top twenty-five kinases. Two volcano plots were generated where the first volcano plot shows the spread of the data for all phosphosites in the uploaded file after filtration. The second volcano plot shows the spread of the data for all phosphosites which have an identified kinase. To produce the volcano plots, the p-value data and fold change data needed to be log transformed, as the log of zero produces an error. The fold change values of zero were removed prior to being log2 transformed while log10 was used for the p-value.

In addition to the graphs, the user is able to download three CSV files. The first of the files is the data frame containing: the identified kinases, their control mean, inhibition mean, relative kinase activity and relative inhibited kinase activity, sorted by descending relative kinase activity. The second file is comprised of the user's uploaded file after the initial data filter with an appended column of the identified kinases. This could be particularly useful if the user wishes to do further analysis on their data. The third file is made up of the same information as the second file, but has the blank kinase values removed.

7. Deployment:

7.1. **Web**

Amazon Web Services (AWS) (15) contains more than one hundred services. You can get to use AWS for free and test it out for one-year free of cost. AWS is also highly reliable and has advanced security features because our information will still be safe even if a natural disaster strikes one of the data centres. Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. The language we used was Python. In terms of deployment, we can choose Flask or Django in Python platform. Our group generated a Flask application and deployed it to an AWS Elastic Beanstalk environment. It is quite simple to deploy our web application using AWS Elastic Beanstalk because you just upload the prepared files and Elastic Beanstalk automatically handles the deployment and then it will create an appropriate environment in the application.

In terms of the prepared files, we prepared all documents for the website, which included py files, CSV files, TSV files, HTML files, and requirements.txt. There were the packages and the version of packages in requirements.txt, which gave the AWS system an instruction to install the required packages in the created virtual environment. There are two ways to create requirements.txt. One way is to create an empty text file and then type the packages and the required version. The other way is to use the terminal.

7.2. **AWS Elastic Beanstalk**

We opened the AWS Elastic Beanstalk website (log in AWS count, and searched “Elastic Beanstalk” in “Services”) and followed the instructions to upload the zip files. In our case, there was an error called “404 not found”. Therefore, we modified the WSGIPath to “main.py” (main.py is the main Python file to run our website) in “Software”, and changed “Ignore HTTP 4xx” to “enable” to ignore the “not found” error. All modifications were made in “Configuration”. After modifying, we clicked on “Apply configuration” at top right to refresh the environment. The URL at the top was the successful deployment link.

7.3. **Terminal**

We used the AWS Elastic Beanstalk website to deploy the website.

Notice that if you type command (pip freeze | grep -v "pkg-resources"), there are all installed packages in the requirements.txt, which also contains some extra packages that are not needed for this website development. Thus, if there is to use the terminal to create requirements.txt, the best advice is to create a new virtual environment and work in there.

7.4. Locally

Firstly, we installed the packages that we need in the terminal on a personal laptop. The next step was to run Python files by using Python3, note that "main.py" was the last file to run. And then there was a local link which looked like "http://127.0.0.1:5000/". The link was our website and we could check the functionality of the website.

8. Limitations:

8.1. Data Collection:

- **Standardization of Data Nomenclature**: Naming of kinases, substrates and inhibitors differed across the different sources where we obtained our data. This mainly affected our connectivity between our Kinase and Inhibitor tables as Kinase and Phosphosite data was verified by a common source (UniProt). This led to a few database construction issues later down the line, which required the inhibitor data to be drastically reformatted in order to work in our database.
- **Limited sources of data:** Each of the main three tables of that database were only supported by 2-3 sources each. With many more sources of data available, our database in its present state does not encapsulate all of the data on kinase, phosphosites and inhibitors.
- **Clinical Trial Kinase Inhibitors only**: PKIDB only takes into account Kinase Inhibitors that have either passed clinical trials or are currently within clinical trials. With a number of predicted kinase inhibitor structures being available for researchers to observe, not including this data severely limits the capabilities of our inhibitor section of our database.
- **No connection between Inhibitor table and Phosphoproteomic tool:** Due to a lack of Inhibitor alias/synonyms in the Inhibitor table of our database, there was no possibility for our database to identify the kinase inhibitor used in a user's uploaded data file.

8.2. Web Application:

- **Optimization of user experience:** A number of areas in using our web application still require optimization in order to make the user experience as smooth and as intuitive as possible. An example of this is using the NCBI Sequence Viewer as part of our Phosphosite search, which does not automatically import the Chromosome Location from a search result directly into the viewer.
- **Download issues for Phosphoproteomic Data Analysis:** Due to static files, on the web application being used as templates to run the download capabilities of our analysis tool. If multiple users use the tool at the same time with different files

and download the results, the file will reflect the result of the last uploaded file, instead of producing separate results.

- **Loading time of Phosphoproteomic Data Analysis:** The running time of the phosphoproteomic data analysis tool presently takes 1-2 minutes, however it could be quicker to improve user experience.
- **Only data from one inhibitor data is processed at a given time:** The analysis tool only reads the first 7 columns of the data file, which equates to the data between the control and sample treated with one inhibitor. Therefore, the data for only one inhibitor can be analyzed at any given time, with data files containing data from multiple inhibitors not being fully read and analyzed.
- **Limitation to Relative Kinase Activity calculation:** Kinase activity analysis (KAA) method has limitations as it relies on computational predictions of kinase-substrate relationships, which can be susceptible to errors. Also, the kinase activity is dependent on the number of detected targets and the fold change between conditions is not taken into account. This is a limitation because not all phosphosites are detected so the calculated relative kinase activity may not be reliable (27).

9. Future Developments:

Possible ideas to further develop Kincyclopedia in the future can be split into three categories:

- **Improving current features:**
  - Improving the usefulness and interpretability of the plots, Kincyclopedia currently generates by implementing things such as standard deviation error bars and formatting the current plot to highlight differences between kinase relative activity vs. inhibited activity in the bar plot for clearer result interpretability.
  - Allowing for multiple users to use the phosphoproteomic data analysis tool, and download the output files simultaneously with no errors.
  - Including other types of kinase activity calculations to overcome limitations in the KAA method. In the future, a method such as Kinase-Substrate Enrichment Analysis (KSEA) can also be used as this generates kinase activity scores using log2-transformed fold change data between conditions (initial and treated states) (30).

- **Developments for long-term scalability of web application:**
  - As SQLite3 is designed to be a lightweight relational database management system, if we plan to expand the functionalities of Kincyclopedia in the future. This could require expansions to the current schema structure and also requiring more data storage. With SQLite3 having its own data storage limits, to make these extra functionalities a reality creating the database using another management system, such as MySQL (31), would be beneficial in the long term.
  - With new research results likely to emerge every day, which could result in new kinases, phosphosites or inhibitors being discovered. Researchers may want to upload their new relevant data to our web application to keep the database records updated. So providing a feature to allow just this is would also prove to be useful in terms of scalability.

- **New features:**
  - Capability for researchers to search for kinases or substrate by amino acid or nucleotide sequences using a BLAST function similar to one KinBase (2) has.
  - Ability for users to view kinase-kinase and kinase-substrate interactions visually by implementing and embedding packages and programs such as the UniProt Interaction Viewer (32) and Cytoscape (33).
  - Allowing the user to search by disease associations based off of a search kinase and as well as its activity. Something similar to this has already been implemented as part of KinMap (17), so in theory embedding/integration of this feature should be possible.

10. References:

1. Eid S, Fulle S. KinHub [Internet]. [cited 2020 Feb 12]. Available from: http://kinhub.org/kinases.html
2. Manning G, Whyte DB, Martinez R, Hunter T, Sudarsanam S. The Protein Kinase Complement of the Human Genome. Science. 2002 Dec 6;298(5600):1912–34.
3. The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019 Jan 8;47(D1):D506–15.
4. Carles F, Bourg S, Meyer C, Bonnet P. PKIDB: A Curated, Annotated and Updated Database of Protein Kinase Inhibitors in Clinical Trials. Molecules. 2018 Apr;23(4):908.
5. Hornbeck PV, Zhang B, Murray B, Kornhauser JM, Latham V, Skrzypek E. PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. Nucleic Acids Res. 2015 Jan 28;43(D1):D512–20.
6. Hipp DR, Kennedy D, Mistachkin J. SQLite [Internet]. 2015 [cited 2020 Feb 12]. Available from: https://www.sqlite.org/docs.html
7. Bayer M. SQLAlchemy [Internet]. 2006 [cited 2020 Feb 12]. Available from: https://docs.sqlalchemy.org/en/13/
8. The Pallets Projects. Flask-SQLAlchemy [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://flask-sqlalchemy.palletsprojects.com/en/2.x/
9. WTForms Team. WTForms [Internet]. 2018 [cited 2020 Feb 12]. Available from: https://wtforms.readthedocs.io/en/stable/
10. The Pallets Projects. Flask [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://flask.palletsprojects.com/en/1.1.x/
11. Mozilla Developer Network. DevDocs - HTML [Internet]. 2018 [cited 2020 Feb 12]. Available from: https://devdocs.io/html
12. Mozilla Developer Network. CSS: Cascading Style Sheets [Internet]. MDN Web Docs. [cited 2020 Feb 12]. Available from: https://developer.mozilla.org/en-US/docs/Web/CSS
13. Otto M, Thornton J. Bootstrap Documentation [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://getbootstrap.com/docs/4.1/getting-started/introduction/
14. Mozilla Developer Network. DevDocs - JavaScript [Internet]. 2018 [cited 2020 Feb 12]. Available from: https://devdocs.io/javascript
15. Amazon Web Services, Inc. AWS Elastic Beanstalk [Internet]. What is AWS Elastic Beanstalk? - AWS Elastic Beanstalk. [cited 2020 Feb 12]. Available from: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
16. McKinney W. Data Structures for Statistical Computing in Python. In 2010 [cited 2020 Feb 12]. p. 51–6. Available from: https://conference.scipy.org/proceedings/scipy2010/mckinney.html

17. Eid S, Turk S, Volkamer A, Rippmann F, Fulle S. KinMap: a web-based tool for interactive navigation through human kinome data. BMC Bioinformatics. 2017 Jan 5;18(1):16.
18. Reitz K. Requests: HTTP for Humans™ [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://requests.readthedocs.io/en/master/
19. Richardson L. Beautiful Soup Documentation [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
20. The Python Standard Library. csv — CSV File Reading and Writing — Python 3.8.2rc1 documentation [Internet]. [cited 2020 Feb 12]. Available from: https://docs.python.org/3/library/csv.html
21. Piacentini M. DB Browser for SQLite [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://sqlitebrowser.org/dl/
22. National Center for Biotechnology Information (NCBI). NCBI Sequence Viewer 3.34.0 [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://www.ncbi.nlm.nih.gov/projects/sviewer/
23. European Molecular Biology Laboratory - European Bioinformatics Institute. Nightingale - re-usable visualisations for the life sciences [Internet]. 2020 [cited 2020 Feb 12]. Available from: https://ebi-webcomponents.github.io/nightingale/#/
24. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, et al. The ChEMBL database in 2017. Nucleic Acids Res. 2017 Jan 4;45(Database issue):D945–54.
25. Ardito F, Giuliani M, Perrone D, Troiano G, Lo Muzio L. The crucial role of protein phosphorylation in cell signaling and its use as targeted therapy (Review). Int J Mol Med. 2017 Aug 1;40(2):271–80.
26. Veredas FJ, Cantón FR, Aledo JC. Methionine residues around phosphorylation sites are preferentially oxidized in vivo under stress conditions. Sci Rep. 2017 Jan 12;7(1):1–14.
27. Wirbel J, Cutillas P, Saez-Rodriguez J. Phosphoproteomics-Based Profiling of Kinase Activities in Cancer Cells. In: von Stechow L, editor. Cancer Systems Biology: Methods and Protocols [Internet]. New York, NY: Springer; 2018 [cited 2020 Feb 12]. p. 103–32. (Methods in Molecular Biology). Available from: https://doi.org/10.1007/978-1-4939-7493-1_6
28. Qi L, Liu Z, Wang J, Cui Y, Guo Y, Zhou T, et al. Systematic Analysis of the Phosphoproteome and Kinase-substrate Networks in the Mouse Testis. Mol Cell Proteomics MCP. 2014 Dec;13(12):3626–38.
29. Bokeh Development Team. Bokeh 1.4.0 documentation [Internet]. 2019 [cited 2020 Feb 12]. Available from: https://docs.bokeh.org/en/latest/docs/installation.html
30. Casado P, Rodriguez-Prados J-C, Cosulich SC, Guichard S, Vanhaesebroeck B, Joel S, et al. Kinase-Substrate Enrichment Analysis Provides Insights into the Heterogeneity of Signaling Pathway Activation in Leukemia Cells. Sci Signal. 2013 Mar 26;6(268):rs6–rs6.

31. Oracle Corporation. MySQL :: MySQL Documentation [Internet]. 2019 [cited 2020 Feb 13]. Available from: https://dev.mysql.com/doc/
32. European Bioinformatics Institute - UniProt. ebi-uniprot/interaction-viewer [Internet]. UniProt (EBI); 2019 [cited 2020 Feb 13]. Available from: https://github.com/ebi-uniprot/interaction-viewer
33. The Cytoscape Consortium. Cytoscape 3.7.2 User Manual — Cytoscape User Manual 3.7.2 documentation [Internet]. 2019 [cited 2020 Feb 13]. Available from: http://manual.cytoscape.org/en/stable/