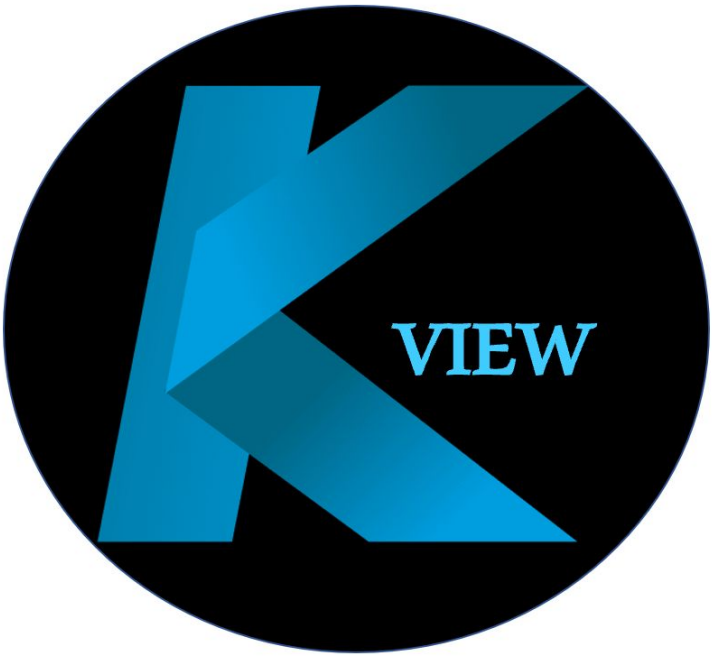


Documentation

Produced by:

Arslan Hussaini, Maria Jafarpour, Pedro Moreno Cardoso and Yuting Shu

About **3**

Software Architecture **4**

Website architecture **5**

Running kinview **6**

Data mining **6**

Kinase data 7

Substrate and phosphosite data 7

Inhibitor data 7

Database **8**

SQLite 9

Structure and tools used within website **9**

Deployment **10**

Website functions **11**

Home page 11

Kinase search 11

Genome Browser 11

Kinase activity upload 12

Phosphoproteomics data analysis 12

More information links 13

Error page 13

Limitations **13**

Data limitations 13

Database limitations 14

Website usability limitations 14

Future developments **15**

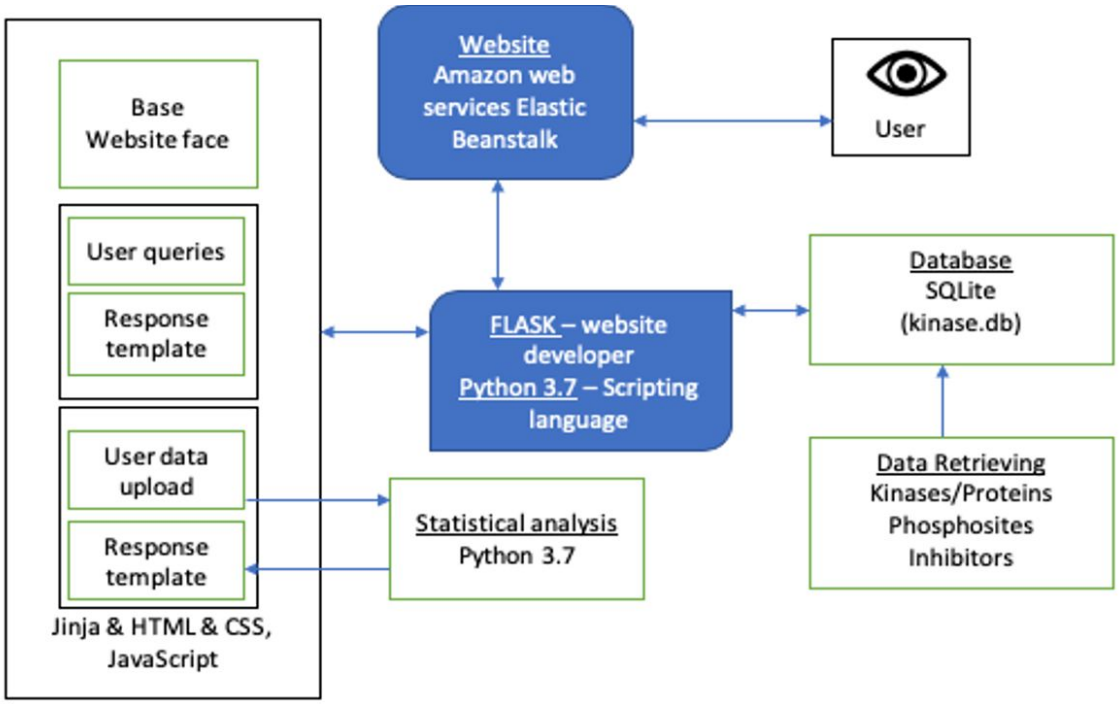
References **15**

About

Kinases play an important role in Biology and Biochemistry and as such a tool to browse knowledge about the various human kinases, their targets and their inhibitors would prove to be invaluable. As such, we have developed a web application dedicated to providing information about human kinases. This has been developed by four MSc Bioinformatics students at Queen Mary, University of London under the supervision of Professor Conrad Bessant and Dr. Fibrizio Smeraldi.

Kinview is a webapp for finding information about human kinases, their inhibitors, the substrates and the various phosphosites on these substrates. A genome browser is also available to browse the phosphosites present in each chromosome. In addition a tool for analysing Phosphoproteomic data and returning kinase activity levels is also available. This web application was developed as part of the MSc Bioinformatics course at Queen Mary, University of London.

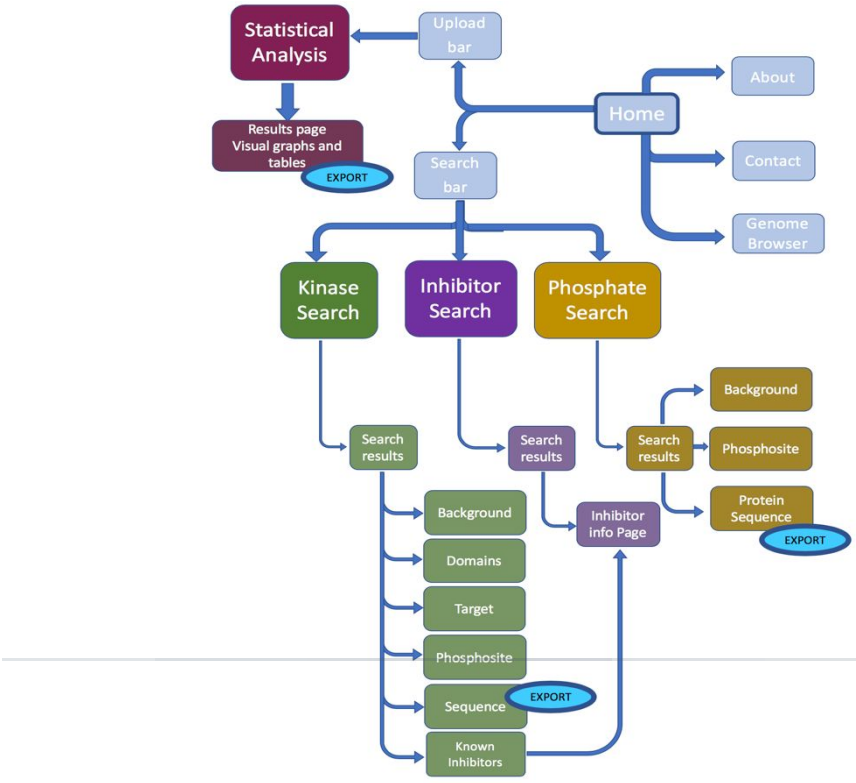
Software Architecture



The main components of our website are shown in Figure 1. Tables were created with all necessary information on the kinases, phosphosites and inhibitors. All this information was integrated into a database file called “kinase.db” using SQLite code. SQLite connects the database with main application Flask. With SQLite queries we can easily and quickly access the necessary data from the “kinase.db” file. Flask incorporates the front-end of the website using Jinja as template engine and Python 3.7. The main Flask files used are: (1) application.py, which is the executable file; (2) DataAccess.py, responsible for the integration of the database; (3) Routes.py, responsible for the different website pages; (4) analysis.py, which is responsible for analysing and creating the results of the file uploaded by the user. All website routes are defined by using HTML language, static files written in CSS were also created to maintain a constant design and also used Javascript language for some visualization aspects. All these files are interlinked with the main application Flask. The interconnection of all these separate parts return the final, fully functional website. We then deployed our website application using Amazon Web Services Elastic Beanstalk making the website available for any user.

Website architecture

The architecture of this web application is shown here in Figure 2. Here is showing the routes linking each part of the web application and their related links with each part separated by colour coding. This concise summary shows the journey from home page through every part of the application. The Home pagehas three top bar tabs for about, contact and genome browser. In the centre of the home page there is one direct route to three search engines and one direct link to a data analysis tool. Each search tool outputs a page with information allocated into different tabs. The export functionalities allow the user to save protein sequences and statistical analysis results for future reference.



Running kinview

Kinview is deployed on Amazon Web Services Elastic Beanstalk and can be accessed from the following link:

http://kinview-env.p3rqxnfk29.us-west-2.elasticbeanstalk.com/

If you wish to run Kinview locally, python 3.6 or higher and git must be installed. First, the git repository must be cloned, and then you must navigate into the cloned repository. Then the following commands must be run:

For Mac OS and Linux:

1. python3 -m venv venv
2. . venv/bin/activate
3. pip install -r requirements.txt
4. export FLASK_APP = "application.py"
5. flask run

For Microsoft Windows:

1. pip3 install -r requirements.txt
2. $env:FLASK_APP = ”application.py”
3. flask run

Data mining

To retrieve the data, we had four primary sources:

- Uniprot: a database of proteins, containing a variety of functional and structural information
- European Bioinformatics Institute which also contains a variety of information on many proteins in an easily parsable XML format
- Ensembl which provides genomic information on a number of proteins and their transcripts
- PhosphoSitePlus: a database of post-translational modifications in human, mouse and rat proteins

The data mining was completed via python scripts and the various APIs available on the respective sources. The data was incorporated into a number of csv files, which could then be used to populate our database.

Kinase data

For our database we required a variety of information on human kinases. We retrieved a list of human kinases from the uniport database (https://www.uniprot.org/docs/pkinfam) and used this list to decide which proteins we needed to obtain information for. We then use the uniport API (https://www.uniprot.org/help/api) as well as the European Bioinformatics Institute protein data bank (https://www.ebi.ac.uk/pdbe/node/1). With EBI, we were able to get protein information in an XML format which could then be parse using the BeautifulSoup package in python to retrieve the relevant information. This information included the gene and protein names for each kinase, the domains and positions, the phosphosites on these kinases as well as the protein sequence for each kinase.

Substrate and phosphosite data

Substrate and phosphosite data was taken from PhosphoSite Plus. More specifically, the kinase-substrate dataset (https://www.phosphosite.org/staticDownloads). The data was then filtered to only include phosphorylation of human proteins. This data provided us with a list of substrates and phosphosites as well as information about the kinase, the kinase accessions, the substrates, the substrates accessions and locations of phosphorylation.

Inhibitor data

Our data for the Inhibitor database is sourced from the inhibitor data page of the website "International Centre for Kinase Profiling" (http://www.kinase-screen.mrc.ac.uk/). It contains a variety of information about a number of inhibitors, including name, Cnumber, molecular formula , structure (SMILES), molecular weight and the various targets of these inhibitors, which is shown by experimental data showing remaining kinase activity under the influence of the inhibitor.

The aliases of the inhibitors are also contained within our database, taken from ChemSpider. The chemical structures of the inhibitors were genearated using the SMILES formula using the tool "Open Babel" which is available as part of the python anaconda package.

We then filtered the relevant data using python and made a number of .'csv' tables according to our established data schema. The generation of tables was primarily done through the 'pandas' package in python.

Database

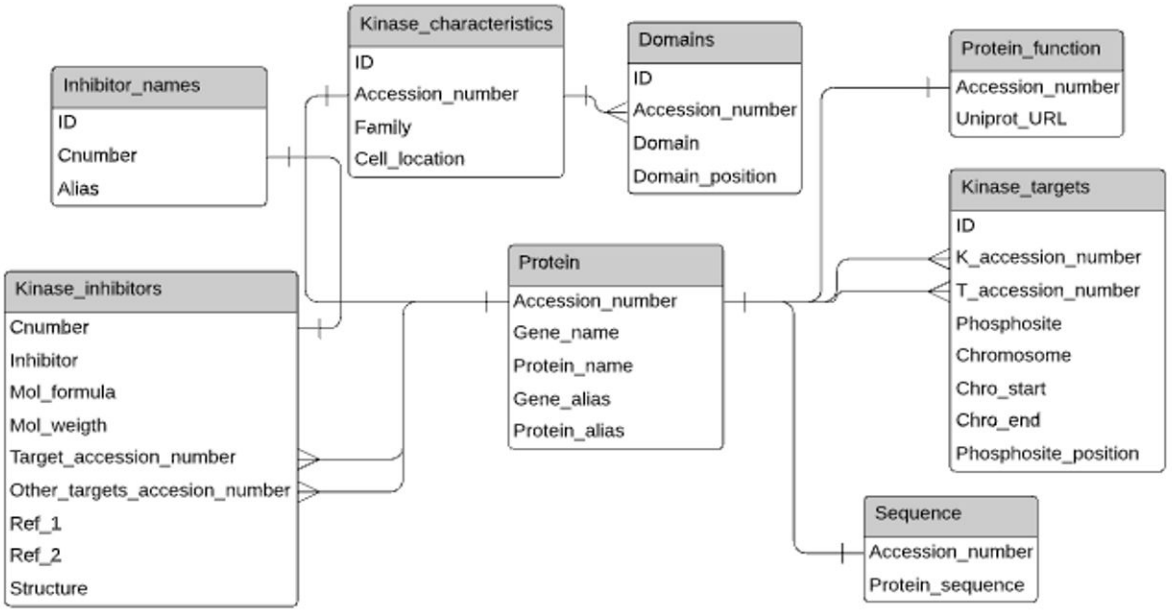


Figure 3: Kinview Database schema. The figure shows the different tables included in the database and the lines represent the links between the tables. Also, all tables where exists a column with accession_number can be linked with any other table that has access

The database schema was designed having in mind the information that needed to be displayed on the web application. The tables shown in figure 2 were created individually and through SQLite, the database was generated and populated. The links between tables are the Accession_number. In table Kinase_targets, K_accession_number refers to the kinase accession number and the T_accession_number is the target of the kinase, also an accession number. Having the accession number as a link in all the tables allows us to quickly retrieve information from any table if we have the accession number associated with the protein we are interested. The Inhibitor_names table shows all the names of a specific inhibitor and its assigned Cnumber. This Cnumber is then used to link this table with the Kinase_inhibitors table. This table is then linked with the rest of the database by having the accession number of the targets and other targets of each inhibitor. For example, if the user searches for protein X, we search X on the Protein table (Protein_name, Gene_name, Gene_alias or Protein_alias columns) to get the accession number assigned to that protein. From there we can get the kinases that affect protein X and the proteins that X might target by assessing table Kinase targets.

If we would like to know which inhibitors target protein X, we just need to look for the Protein X accession number in the columns Target_accession_number and Other_target_accession_number from the table Kinase inhibitors and extract the column Inhibitor when the accession number was found in the target columns. The schema might look that it could have less/joined table but the reason for this is that the table Protein contains all existing human proteins, while other tables, like for example Kinase characteristics only has information for proteins that are actually kinases. Another reason is that, some tables have the accession number as a primary key (each accession number only appear once) and other tables such as the Domains table have several entries for the

same accession number. We could also separate the kinase targets into two tables, one with kinases and other with targets/substrates. However, we found that having the tables this way reduces the number and the complexity of the queries necessary to extract both phosphosites and kinases of a determined protein.

SQLite

For the generation of our database and the connecting of our database to Flask we chose to use SQLite. The primary reason for this choice over other options such as MySQL was primarily due to SQLite not requiring a client-server engine. This simplified our architecture and allowed us to easily integrate the database into our website and deploy it through AWS Elastic Beanstalk. For our relatively small database, SQLite serves us adequately and allows for fast querying and loading of data. Our database is also static, meaning the major drawback of SQLite which is that multiple people cannot be writing to the database, is not relevant in our design.

Structure and tools used within website

HTML/CSS:

Bootstrap 4: This is used for general look and feel of the web application. Bootstrap is widely used for styling of websites. We used it for styling our HTML elements and div(s). For example: to create a HTML page in several parts we used cards to display each part, allowing us to implement an information-related division on page.

HTML tags: These were used to define the components of each page. HTML was coupled to CSS languages to deliver a well presented application front with visual features for a more enjoyable experience.

CSS was used to generate professional search fields, and to work with the JavaScript and other interactive elements. For example: CSS grid was used to ensure separate cards on the page are in line. Other format and presentation aspect were also implemented using CSS

Limitations We could have looked into using W3-CSS is a CSS framework that also contains built in responsiveness. It is smaller and faster than CSS and with a few commands produced a clean result. Furthermore, this would eliminate the need to manually update CSS.

JavaScript:

Phosphokinase runs opensource JavaScript libraries to provide an interactive experience for the user, as well as providing another platform for more detailed information about protein kinases. JavaScript libraries that we have used are:

jQuery: jQuery is a widely used JavaScript library for introducing interaction to the website. JavaScript handles all the client-side dynamism in web applications. jQuery works as a library for

very commonly used web interactions. Bootstrap also uses jQuery. We used it for implementing export functionality.

html2canvas: This is being used as a tool to convert the html page to a html canvas image. This image could be later used to create the pdf that is exported.

Jspdf: This is a library that we are using to create the pdf from the canvas image.

Python:

os: This is the python library that we use for general os variables, like current working directory and creating file paths.

flask: Flask is a light-weight web application framework that helps us build dynamic web applications with least footprints. We have used several flask extensions to handle forms, errors and variables**.** Flask is based on Jinja 2 templating (Jinja2 works with Python 2.6.x, 2.7.x and >= 3.3) which shares a language similar to python, this made it easy to integrate and run code on the main application as well as on the HTML pages.

werkzeug: We use this library to handle file names. It was necessary to handle file names while uploading the files to web app.

pandas: pandas is used for reading data from csv or other data files. We have used it in Kinase Analysis and in database Access.

numpy: numpy is widely used for creating graph plots we have used this in Kinase Analysis as well.

bokeh: It is a modern library that helps create interactive presentation in modern web browsers, we used it for plotting the graphs in Kinase Analysis.

Deployment

To deploy our website, we chose to use Amazon Web Services Elastic Beanstalk Free tier. We chose this service as much of the server configuration is automatic upon setting up your environment. This allowed us to focus on other areas, rather than server configuration and system administration work. Despite this automation however, Elastic Beanstalk is still very customisable which would allow us to modify it to fit our needs in the future if we ever need to do so. Elastic Beanstalk does come with its limitations however namely in speed of deployment and failures in deployment where sometimes it is not clear what has failed, and Elastic Beanstalk provides little in the way of indications of what caused this failure.

Website functions

Home page

The web application has been designed to give the user a smooth experience flowing through out the website, connecting all parts without reaching any dead ends. From the home page all searches are displayed according to their relevance. The central based search is connected to our database backend. The kinase activity upload is connected to analysis python codes relevant for kinase activity analysis of .tsv files. At the top the genome browser is linked to an external website NCBI, which allows the user to search the whole human genome using a known database outside our own one. Also, tabs at top bar present contact and about information pages for the user to get in touch with the development team if necessary, for enquiries or collaboration on a more personal level. The link of the home page is implemented throughout the website, allowing the user to go back to start point at any stage through their experience.

Kinase search

Using a case insensitive search, user is able to search our database through selecting from our dropdown options (Kinase name, inhibitor name, phosphate name). This leads to a search page relevant to the closest match to user enquiry. User can select the search that they prefer and then see the page of information related to that selection. User then sees a page of information in either tabs format or single page. If in tab format, each tab displays the relevant information of aspects related to user enquiry search. The protein sequence of the kinase can be exported for future reference or external use. The inhibitors targeted by that kinase are linked to their own information pages. If the user wishes to see more information related to the inhibitor, the user can click as intended.

Genome Browser

To allow users to browse phosphosites by their genomic location, we incorporated a custom embedding of the NCBI sequence viewer. Our genome browser allows users to select a chromosome to view and the viewer will display the genomic sequence of said chromosome with markers placed at phosphosites within the chromosome. The markers are named with the accession of the phosphosite along with the amino acid residue phosphorylated and the position of said amino acid residue within the protein. Also available on our sequence viewer are the six reading frames for the genome.

Kinase activity upload

The user is able to upload statistical data in form of tsv file to get an interactive output of statistical analysis consisting of quality of the image to be rendered which is then displayed in the browser. The user is then able to click the export link which results in an option to download the image as a PDF file to the user's computer.
If an incorrect filetype is uploaded, the software rejects the file and displays an error message. Further analysis could be considered for further development. The size of file upload is also another limitation. The library modules used for rendering the statistical results are currently only works with 32bit of python. Unfortunately, using 32bit python would cause memory errors when trying to upload large files (> 300MB) causing the program to crash. This was an issue which currently has not been resolved and is important to discuss as the creation of software is meaningless if it has not been coded for different platforms.

Phosphoproteomics data analysis

The website provides a tool called Kinase activity analysis that allow the user to upload phosphoproteomic data and receive a summary of the uploaded data, a volcano plot and the relative human kinase activity. The user must upload a .tsv file with only one inhibitor and the columns should be presented in a specific order: Substrate, Control mean, inhibitor mean, fold change, p-value, control CV, inhibitor CV. The substrate column accepts both protein names and gene names.

After the file is uploaded the data is processed. The data is processed in a series of steps. First, we provide a bar graph summarizing the data present in the file. In the bar plot we show the total number of substrates and the number of substrates where the columns are empty. Furthermore, out of the substrates that were not empty we show how many of them have significant fold changes. From the significant ones, we show the number of substrates that showed increased or decreased phosphorylation.

On the second step we completely clean the data. We remove all columns that are completely empty and remove any row that has at least on value empty. After this a volcano plot is produced. This volcano plot shows all substrates with two different colours. The yellow dots are the substrates that show a low fold change (less than 100 and bigger than 0.1) and the blue dots that are the one with the higher level of fold change. In the volcano plot it is also possible to see a red horizontal dashed line that sets the p-value on 0.05. The x and y axis of the plot show the log10 of both the fold change (x-axis) and the p-value (y-axis).

The third step is to calculate the relative activity of the kinases. For this we used the KSEA algorithm described by Casado et al., 2013. With this algorithm we calculate a kinase z-score as shown in figure 3 below.

$$score = \frac{\left(\bar{s} - \bar{p}\right)\sqrt{m}}{\delta}$$

Figure 3: KSEA algorithm. $\bar{s}$ - denotes the mean log2(FC) of known phosphosite substrates of the given kinase, $\bar{p}$ - represents the mean log2(FC) of all phosphosites in the dataset, *m* denotes the total number of phosphosite substrates identified from the experiment that annotate to the

specified kinase, and δ denotes the standard deviation of the log2(FC) across all phosphosites in the dataset.

To find the kinases that were affecting the substrate we used the curated Kinase–Substrate annotations from PhosphoSitePlus (Hornbeck P.V et al., 2012), which was also used to retrieve data for one of the tables of our database. The z-score obtained after all the calculations is based exclusively on the phosphorylation status of the substrates. If the z-score is negative means the kinase is being inhibited, if positive, then the kinase is upregulated. Then the p-value is obtained assuming that the z-score follows a normal distribution and that the standard deviation is 1 and the mean 0. The results a bar graph that shows all kinase z-scores and a table below containing all kinases, their z-score and p-value. This bar plot is interactive and allow the user to zoom in and see the exact kinase and z-score associated with it. The number of substrates that didn´t have any kinase matched are also given. The tool also provides the kinases the have a significant p-value (<0.05) of the z-score by showing another interactive bar graph.

S^{-} denotes the mean log2(FC) of known phosphosite substrates of the given kinase, p^{-} represents the mean log2(FC) of all phosphosites in the dataset*, m* denotes the total number of phosphosite substrates identified from the experiment that annotate to the specified kinase, and δ denotes the standard deviation of the log2(FC) across all phosphosites in the dataset.

More information links

These links are present throughout the tabs where necessary for the user to see relevant websites such as Uniport of NCBI for more information such as 3D images.

Error page

Message uploads letting the user know that they have reached an unsatisfied search and the user is directed via a link back to the home page.

Limitations

Data limitations

Our data is somewhat limited in the sense that there is some vital information missing. For example, for the phophosites of a given protein, there are many cases where we have no information what kinases are responsible for this phosphorylation. It is possible that this data can be found from other sources, or potentially from further experiments in the field. Also due to the time constraints in developing this webapp, there is also information missing about kinases and substrates, such as their functions or if they are associated with any diseases.

Our inhibitor data is also limited. We are missing many inhibitors again due to time constraints and also the sources we used. If provided further time to develop we would like to increase the size of our inhibitor database as well as provide further information about these inhibitors.

We have a limited number of inhibitors in our database, as well as some information missing for the inhibitors we do have such as chemical structure and some aliases. There are also issued of chirality with inhibitors.

Database limitations

Our database schema runs fast and correctly. However, if we were to increase drastically the amount of data, we have in the database that would probably make our website much slower. In that case the schema should be changed and look up tables should be added to decrease searching time.

SQLite is also somewhat limiting in the sense that it is not the best tool when needing to update the database. For that reason, it may be a good idea to shift the database to a server based SQL framework such as MySQL.

Website usability limitations

There were several limitations in smooth usability for searching, such as user having to select a drop down in the search rather than having a combined search. Furthermore, while the user is searching a drop down of relevant accusable options do not become present to help navigate the user through their typing.

This application has been designed for website access and is not compatible for a phone app. Therefore, users do not have the easy accessibility of downloading our application onto their smart phones for quick access. They must instead go onto our website, taking more time to do their search.

Interactive 3D images of kinases are not presented; thus, the user has to click the link provided and further search an external website for that image. This would take the user attention away from our application.

These limitations slow down the user from getting to their ultimate destination on our application. Therefore, it is important to consider these as future developments. These are vital for our users to successfully reach their query and to keep our users from leaving this application.

Future developments

Our database was developed in a short time frame and is not comparable to that of major biochemistry/sequence websites. However, to develop our application further vital improvements must be made in the fields of several possible improvements, technical solutions / optimisation, general further development

- Interactivity
- Statistical analysis of files link to download graphs
- Ability to Login
- Updatability
- User interface visual experience
- Backend database design and data available
- User personalisation tools to save certain kind of search history or molecule of interest.

References

Casado Pet al. (2013) Kinase–substrate enrichment analysis provides insights into the heterogeneity of signaling pathway activation in leukemia cells. *Sci. Signal.*, 6, rs6–rs6.

Hornbeck P.V et al. (2012) PhosphoSitePlus: a comprehensive resource for investigating the structure and function of experimentally determined post-translational modifications in man and mouse. *Nucleic Acids Res.*, 40, D261–D270.

The UniProt Consortium

UniProt: a worldwide hub of protein knowledge

Nucleic Acids Res. 47: D506-515 (2019)

Hornbeck PV, Zhang B, Murray B, Kornhauser JM, Latham V, Skrzypek E PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. Nucleic Acids Res. 2015 43:D512-20.

Fiona Cunningham, Premanand Achuthan, Wasiu Akanni, James Allen, M. Ridwan Amode, Irina Armean, Ruth Bennett, Jyothish Bhai, Konstantinos Billis, Sanjay Boddu, Carla Cummins, Claire Davidson, Kamalkumar Jayantilal Dodiya, Astrid Gall, Carlos García Giro´n, Laurent Gil, Tiago Grego, Leanne Haggerty, Erin Haskell, Thibaut Hourlier, Osagie G. Izuogu, Sophie H. Janacek, Thomas Juettemann, Mike Kay, Matthew R. Laird, Ilias Lavidas, Zhicheng Liu, Jane E. Loveland, José C Marugán, Thomas Maurel, Aoife C MacMahon, Benjamin Moore, Joannella Morales, Jonathan M

Mudge, Michael Nuhn, Denye Ogeh, Anne Parker, Andrew Parton, Mateus Patricio, Ahamed Imran Abdul Salam, Bianca M Schmitt, Helen Schuilenburg, Dan Sheppard, Helen Sparrow, Eloise Stapleton, Marek Szuba, Kieron Taylor, Glen Threadgold, Anja Thormann, Alessandro Vullo, Brandon Walts, Andrea Winterbottom, Amonida Zadissa, Marc Chakiachvili, Adam Frankish, Sarah E. Hunt, Myrto Kostadima, Nicholas Langridge, Fergal J. Martin, Matthieu Muffato, Emily Perry, Magali Ruffier, Dan M. Staines, Stephen J. Trevanion, Bronwen L. Aken, Andrew Yates, Daniel Zerbino, Paul Flicek

Ensembl 2019.

PubMed PMID: 30407521.

doi:10.1093/nar/gky1113