

DEVELOPMENT OF “KBCOMMONS” – UNIVERSAL INFORMATICS
FRAMEWORK FOR MULTI-OMICS TRANSLATIONAL RESEARCH

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfilment

Of the Requirements for the Degree

Master of Science

By

SIVA RATNA KUMARI NARISSETTI

Dr. Trupti Joshi, Advisor

DECEMBER 2017

The undersigned, appointed by the dean of the Graduate School, have examined the

Thesis entitled

DEVELOPMENT OF “KBCOMMONS” – UNIVERSAL INFORMATICS
FRAMEWORK FOR MULTI-OMICS TRANSLATIONAL RESEARCH

Presented by Siva Ratna Kumari Nariseti

A candidate for the degree of

Master of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Trupti Joshi

Dr. Dong Xu

Dr. Walter Gassmann

For Amma, Naanna, Annayya

ACKNOWLEDGEMENTS

Firstly, I would love to express my wholehearted thanks of gratitude to my advisor, Dr. Trupti Joshi, for her constant guidance and help during my Master's program. I really appreciate her patience, especially her clear and profound explanation when answering my questions. It would not have been possible for me to successfully complete my thesis without her encouragement and support. I am very grateful for the opportunity to join Dr. Joshi's Bioinformatics lab and to conduct research.

I would also like to extend my sincere thanks to my co-advisor, Dr. Dong Xu, for his valuable and graceful advice on our research problem. Interactions with him helped me think critically, especially from a computer scientist perspective. I would also like to specially thank Dr. Walter Gassmann for being on my thesis committee and for his advice on this thesis.

I would like to thank all the Bioinformatics lab members, who were very supportive and helpful. Especially, I thank Shuai Zeng for sharing his academic knowledge and experience and for the great discussions we had when working on research together. I also thank Sadia Akter, Sidharth Sen and Priyanka Basavaraj for providing me with sample data, for their support and friendship.

Finally, I would love to sincerely thank my parents and my brother for their blessings and moral support throughout my life. My brother is my inspiration and all his achievements are my motivation. My parents are my role models for their love and encouragement throughout our education in spite of having a very modest background in a small village of India, where higher education for a girl has only been a dream.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
LIST OF FIGURES AND TABLES.....	v
ABSTRACT.....	viii
1. Introduction	1
1.1 Biological Background.....	1
1.2 Technical Introduction	3
2. Database and Frontend Architecture	8
3. KBCommons Functionalities	17
3.1 KBCommons Flowchart.....	17
3.2 Data Sources and Supported Datatypes	20
3.3 Browse KBCommons.....	22
3.4 Contribute to KB	28
3.5 Add Version to KB.....	30
3.6 Create a new KB	31
4. Case Study	32
4.1 <i>Medicago truncatula</i> Introduction	32
4.2 Create a new KB for <i>Medicago truncatula</i>	33
4.3 Contribute to <i>Medicago truncatula</i> KB	45
4.4 Browse KBCommons for <i>Medicago truncatula</i>	59

4.5 Add version to KB for <i>Medicago truncatula</i>	64
5. Testing	65
6. Applications of KBCommons	68
7. Summary.....	69
8. Future Study	70
9. References	72
VITA.....	77

LIST OF FIGURES AND TABLES

Figure 1. Model-View-Controller cycle	4
Figure 2. A full MVC request cycle for Laravel application	6
Figure 3. Agile Software development life cycle	7
Figure 4. AJAX workflow	9
Figure 5. Organism library metadata	11
Figure 6. Entity-Relation diagram for organisms	12
Figure 7. Metadata for user library	13
Figure 8. Entity-Relation diagram for users	14
Figure 9. KBC home route	17
Figure 10. KBCommons Flowchart	18
Figure 11. KBCommons Home Page	19
Figure 12. KBCHome controller system-1	23
Figure 13. KBCHome controller system-2	24
Figure 14. KBSystemMain view	25
Figure 15. Browse KBCommons organisms	26
Figure 16. Displaying organism's versions	27
Figure 17. Medicago truncatula A17 showing the shoot with leaves and seed pods.....	33
Figure 18. Create a newKB Usecase diagram	34
Figure 19. Create a new organism	34
Figure 20. KBC user sign up.....	35
Figure 21. KBC user log in	36
Figure 22. Javascript dynamic dropdown	37

Figure 23. KBC find organism list.....	37	
Figure 24. Create a new KB for Mtruncatula	38	
Figure 25. Select 6 Essential Files	39	
Figure 26. Less number of essential files	Figure 27. Correct 6 essential files.....	40
Figure 28. Upload 6 essential files	Figure 29. Import 6 files to DB	41
Figure 30. Import 6 files to Database Status.....	42	
Figure 31. Organisms Library table	43	
Figure 32. Mtruncatula DB tables.....	43	
Figure 33. KBCCommons newkb steps-omics option	44	
Figure 34. Contribute to KB workflow.....	45	
Figure 35. Contribute to KB button	46	
Figure 36. Select details for Mtruncatula.....	46	
Figure 37. Omics and Entities data options	47	
Figure 38. RNA-Seq dataset details selection-1	48	
Figure 39. RNA-Seq dataset file uploads-2	49	
Figure 40. 2DGel dataset selection	50	
Figure 41. Bisulphite Sequencing data selection	51	
Figure 42. SNP data selection.....	52	
Figure 43. GeneExpression RNA-Seq table	53	
Figure 44. Differential Expression Cuffdiff table.....	53	
Figure 45. Transcriptomic Microarray selection details	54	
Figure 46. Proteomic MassSpectrometry selection details	55	
Figure 47. Methylation Array selection details.....	56	

Figure 48. miRNA files details	57
Figure 49. Metabolite datatype options	58
Figure 50. Mtruncatula in Browse KB.....	59
Figure 51. Mtruncatula home page	60
Figure 52. Mtruncatula Search Gene ID	61
Figure 53. Mtruncatula gene card page.....	61
Figure 54. Mtruncatula Search Gene ID Chromosomal Information	62
Figure 55.Mtruncatula Search Gene ID Sequence Information.....	62
Figure 56. Differential expression data retrieval	63
Figure 57. Differential expression data displayed in Venn diagram	63
Figure 58. Add version to KB button	64
Figure 59. Add version to KB details	64
Figure 60. SoyKB to KBCommons tools status	71
Figure 61. Timeline of the development.....	71
Table 1. Data types for multi-omics and entities	22

ABSTRACT

Multi-level 'OMICS' data integration for multiple organisms has been one of the major challenges in the era of advanced next generation sequencing and high performance technologies. Biological data has been producing tremendously fast with the availability of these high throughput sequencing technologies at low price and high speed. However, these data are often stored individually across different web resources based on data type and organism, making it difficult to find and integrate them. There are many websites available which store data from different data types and display that data in pie charts or plain text format but limit their data to only one fixed organism. These web-based multi-omics analysis is an efficient and easy way of analyzing the data but it would be difficult for other researchers working with other organisms and with complex data.

The complex multi-omics data requires extensive data management, exhaustive computational analysis, and effective integration to have a one-stop interactive, web-based portal to browse, access, analyze, integrate and share knowledge about genomics and molecular mechanisms, with ultimate links to phenotypes and traits for many different organisms. To achieve this, we have developed Knowledge Base Commons (KBCommons), a platform that automates the process of establishing the database and making the tools available for organisms via a dedicated web resource.

KBCommons is currently supporting four different categories including Plants and Crops; Animals and Pets; Humans and Diseases; Microbes and Viruses. It has four main functionalities including Browse KBCommons, Contribute to KB, Add version to KB, and Create a new KB. Using KBCommons, researchers from different groups with different organisms' data can be shared and accessed among all. KBCommons is an automatic

framework which uses famous and widely used Laravel PHP framework. This is very efficient to deal with complex and diverse biological datasets.

In the Browse KBCommons section, all existing organisms will be displayed under each category and it also shows organisms which can be used as model organisms. KBCommons also displays the logo of each organism along with existing versions, in this way it will give a detailed information on all existing organisms. The user can browse existing data of each organism using various tools including Blast, Multiple Sequence Alignment, Motif Sampler, etc., by going to that particular page. Users can also visualize gene expression and differential expression data via pie charts and plain text.

Add version to KB and Create a new KB are related because of their similar steps in the process, users must bring corresponding data in each section. When a particular organism of interest is not existing then the user can create a new Knowledge Base for that new organism with 6 essential files of Genome Sequence, protein coding sequence for Amino acid, gene coding sequence for Nucleotide and Spliced mRNA transcripts, mRNA sequences in GFF3, and a functional annotation file. In Add version to KB, if an organism is already existing then the user can add a new version to the existing KB with these 6 essential files for the new version.

In Contribute to KB, user can upload multi-omics data including Transcriptomics – RNA-Seq and Microarray; Proteomics – Mass Spectrometry and 2DGel; Epigenomics – Bisulphite Sequencing, Methylation Array, and MBD-Seq Array. We support both gene expression/ protein expression/ or methylation data and differential expression comparison for each data type. We also support different entities including miRNA/sRNA, Metabolite, SNP/GWAS, Plant introduction lines/ Animal strains, and Phenotype/ TRAIT/Diseases.

1. Introduction

Bioinformatics combines Biotechnology and Information Technology. Bioinformatics mainly involves developing and improving on methods for storing, retrieving, organizing and analyzing biodata, whether it be from biotechnology or any other biological science. It requires to have knowledge about the type of biological data we are working on and to be strong at different programming languages including Python, PHP, MySQL, Javascript and many other technologies based on the requirement of the project. In this section, I will explain about the biological background of my research and the technical details of automatic framework and software development life cycle.

1.1 Biological Background

Web-based multi-omics data analysis has become one of the most efficient ways to offer biological researchers to understand their data with less effort and more results. There are various data analysis tools available on the web to process and show data using visualization tools or by giving plain text results. However, they limit their data to only one fixed organism, one such very successful and highly used website is <http://soykb.org/> which is known as Soybean Knowledge Base (SoyKB) [1][2][3]. SoyKB handles storage and integrates different kinds of data including proteomics, transcriptomics, metabolomics, miRNA, SNP, sRNA, and cDNA which is generated by soybean research groups. SoyKB also facilitates users to visualize and analyze data with different tools including Affymetrix Probe ID Mapper, Gene Family Browser and *In Silico* Breeding Program. This makes SoyKB a comprehensive all-inclusive web resource for soybean.

SoyKB has many users from different countries and has been highly cited by many biological researchers who work with soybean organism. Success of tools like SoyKB

gives us confidence and positive directions to research and develop more sophisticated tools with advanced features and functionalities. With this, we are working on extending SoyKB tools and features to other organisms to meet needs of researchers working with other organisms and data.

The advanced next generation and high-throughput sequencing technologies produce multi-level of 'OMICS' data for many organisms at a very fast rate. Raw data without analysis is of not much use and it would be difficult for scientists to make any meaningful insights out of it. To deal with these complex omics data, we need advanced and high performance computational technologies to analyze and integrate for better understanding of the data. There is an increasing need to integrate different types of 'OMICS' data from multiple organisms and to provide efficient ways to analyze multi-levels of multi-omics data.

Towards this, we developed a Knowledge Base Commons (KBCommons) which is comprehensive and interactive web resource for many different organisms belong to Plants, Animals, Diseases, and Microbes. KBCommons shows existing organisms in the Browse Organisms section, facilitates users to create new organism's KB under Create a New organism's KB section, if KB for an organism is existing then it provides users an option to contribute to KB to add new data and user can also add a new version to the existing organism. In this way, researchers with different organisms' data can be shared, viewed and analyzed data with KBCommons very efficiently. This fills the gap between dealing with one organism and multiple organisms. KBCommons is an automatic framework from users uploading data to users accessing data with out requiring any manual processing in between.

1.2 Technical Introduction

An efficient and powerful framework plays an important role when developing web based application to work with complex and heterogeneous biological data. Flexibility and simplicity of the framework is also crucial to add, remove, or update different components as the development is going on and to work with many individuals.

We are using Laravel PHP web framework for our KBCommons development, it is free and open-source. It is created by Taylor Otwell with the intention for the development of web applications following model–view–controller (MVC) architectural pattern [4][5]. MVC is widely used software architectural pattern for implementing user interfaces on computers [6] and it is very simple yet efficient process shown in Figure 1.

Description about MVC, Components, and Interactions:

Model: The model is the central component of the architecture. A model gets a request from the controller and directly manages data, logic, and rules of the application.

View: View can be any output from Model. There can be multiple views of the same information including bar chart, tabular and etc. View shows data to the user.

Controller: It takes input and sends it to the corresponding model or view.

Laravel framework is a fully MVC-compliant framework from 2nd version, Laravel 1 lacked support for controllers [7]. For this development, we are using Laravel 5 [8][9].

Specifically, in Laravel 5 framework I use the following programming languages for each part:

- Controllers to handle user requests and retrieve data, by leveraging Models (PHP+MySQL+Python)
- Models to interact with your database and retrieve your objects' information (PHP+MySQL+Python)
- Views to render pages(HTML+Javascript+AJAX)

Additionally, routes are used to map URLs to designated controller actions.

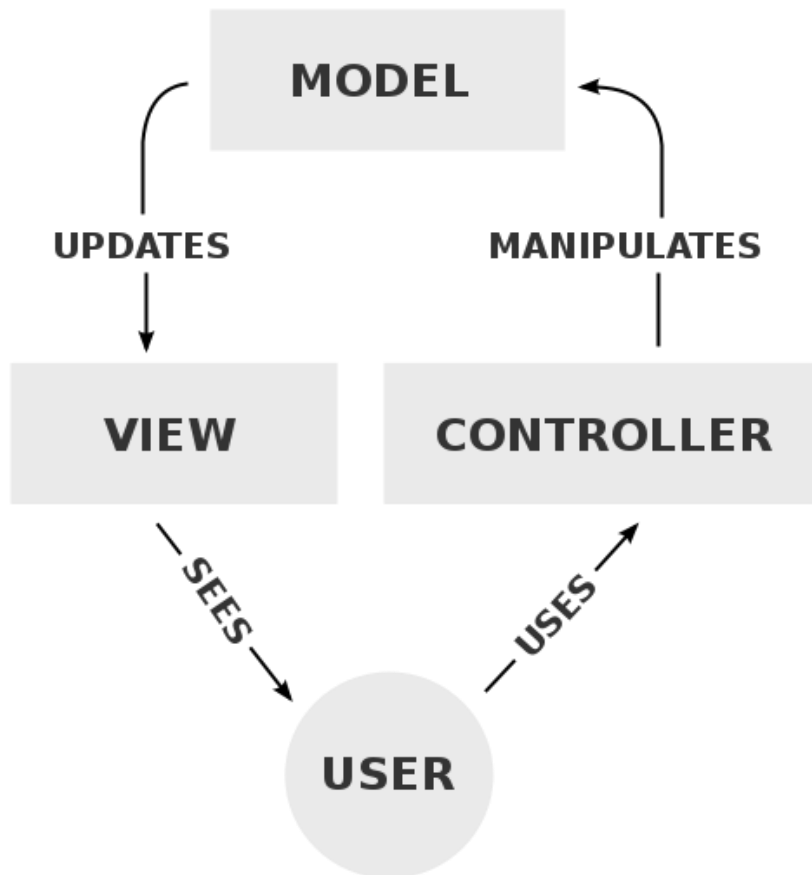


Figure 1. Model-View-Controller cycle

Figure 1 is collected from <https://en.wikipedia.org/wiki/Model-view-controller>

The source code of Laravel is hosted on GitHub and licensed under the terms of MIT License [10]. There are many advantages of using Laravel PHP framework for the development of web-based protocol including the following:

1. **Template engine:** One can build amazing layouts with dynamic content seeding using Laravel's in-built lightweight templates. It provides solid structures which are built using CSS and JS. These templates can be used to build a simple layout with different sections.
2. **Artisan:** Laravel has a built-in tool for 'Artisan' command line and it allows us to avoid many manual repetitive and tedious programming tasks with simple commands. It also gives us a way to generate basic MVC files and skeleton code with their respective configurations.
3. **Libraries:** Laravel has Object Oriented libraries and many other pre-installed libraries, which are not likely found in other popular PHP frameworks. With this feature, it is very convenient to build complex functionalities.
4. **Modular:** Laravel framework is built on more than 20 different libraries and is itself divided into individual modules. It adopts modern PHP principles, which allows developers to build modular, responsive and handy web apps.
5. **MVC Architecture Support:** Laravel supports the MVC pattern, which ensures clarity between logic and visualization. This architecture helps in improving the performance, allows better documentation, and has multiple built-in functions.
6. **Security:** Making sure that applications are secure is very important in any web application. We have to use some or other methods to make the application secure. Another advantage of using Laravel framework is it takes care of the security within

the framework. For generating encrypted password Laravel uses Bcrypt hashing algorithm which uses the very efficient Bcrypt function. Bcrypt is a password hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher, and presented at USENIX in 1999 [11].

A full MVC request cycle for Laravel application is shown below: (collected from <http://laravelbook.com/images/laravel-architecture/laravel-mvc-components.png>)

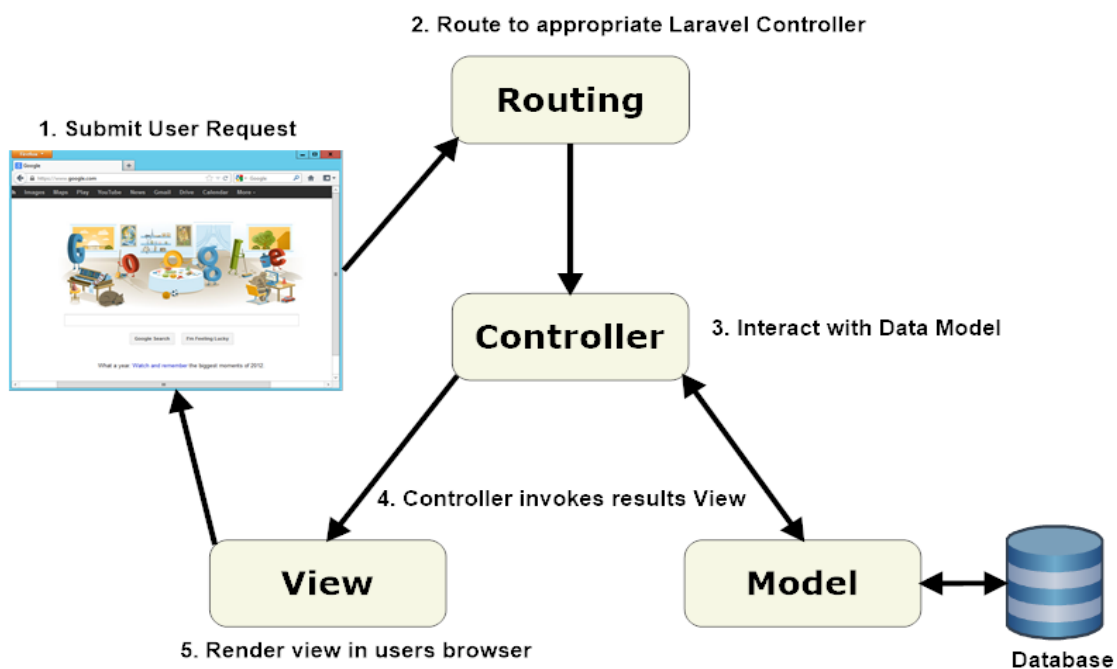


Figure 2. A full MVC request cycle for Laravel application

From the above figure, we can learn that Laravel request cycle has 5 main steps from a user submitting a request to show the appropriate content in the browser for that user request:

1. Submit user request: When user clicks on the required URL, request will be sent to the routers
2. Route to appropriate Laravel controller: In routes, the Laravel controller action will be performed according to the user request

3. Interact with data model: This is the main step, where we retrieve data from the database and send back to controller
4. Controller invokes results view: After obtained results from the model, controller will send the required results to the appropriate view
5. Render view in user's browser: Finally, the required web page along with appropriate data will be displayed in the user browser.

Software Development Life Cycle (SDLC): SDLC is a process of a series of planned activities to develop or alter the software. It is used to design, develop, and test the software to ensure high quality and to reach user needs. There are many kinds of SDLCs including Waterfall model, Iterative model, Spiral model, V-model, Agile model, RAD model, etc., Agile Model is very flexible for small and large projects. It adopts characteristics of incremental and iterative models. It breaks the product into small incremental builds and these can be easily developed by different individuals or teams. In our project we adopted Agile SDLC and our team members work on individual developments which can be delivered to the user directly or combined with other developments.

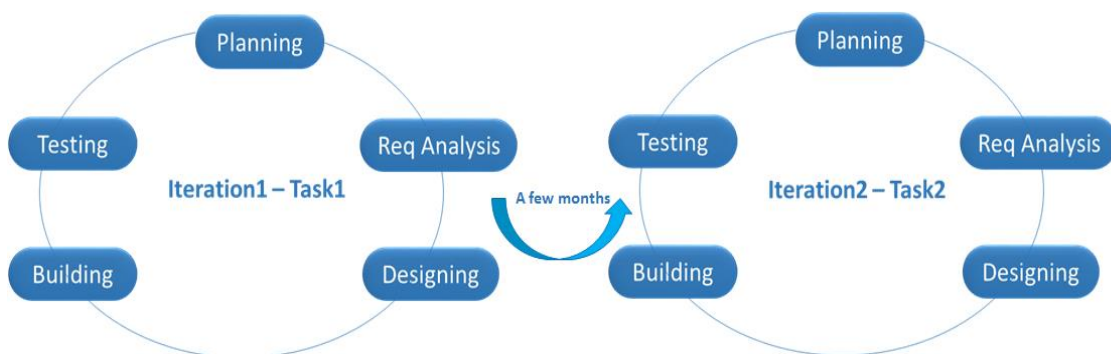


Figure 3. Agile Software development life cycle

2. Database and Frontend Architecture

In this section, I will describe briefly about programming languages that I used and I will explain about database schema and version control importance in interdisciplinary studies where many people work together on the same research project.

In this development, I used PHP, HTML, Javascript, and AJAX for web page development; Python and MySQL for backend and the connection between frontend and backend. We also used GitHub for version control and for collaborating among developers

- **PHP:** is a server-side scripting language originally created by Rasmus Lerdorf in 1994 [12]. It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. PHP is designed to make dynamic pages and applications. PHP supports MySQL, Oracle, Sybase and many other databases.
- **AJAX:** it is Asynchronous JavaScript And XML. AJAX is not a programming languages it just uses a combination of a browser built-in XMLHttpRequest object to request data from a web server; Javascript and HTML to display or use the data. Working of AJAX is shown in the below picture, taken from <http://www.j2eebrain.com/java-J2ee-jquery-ajax.html>.

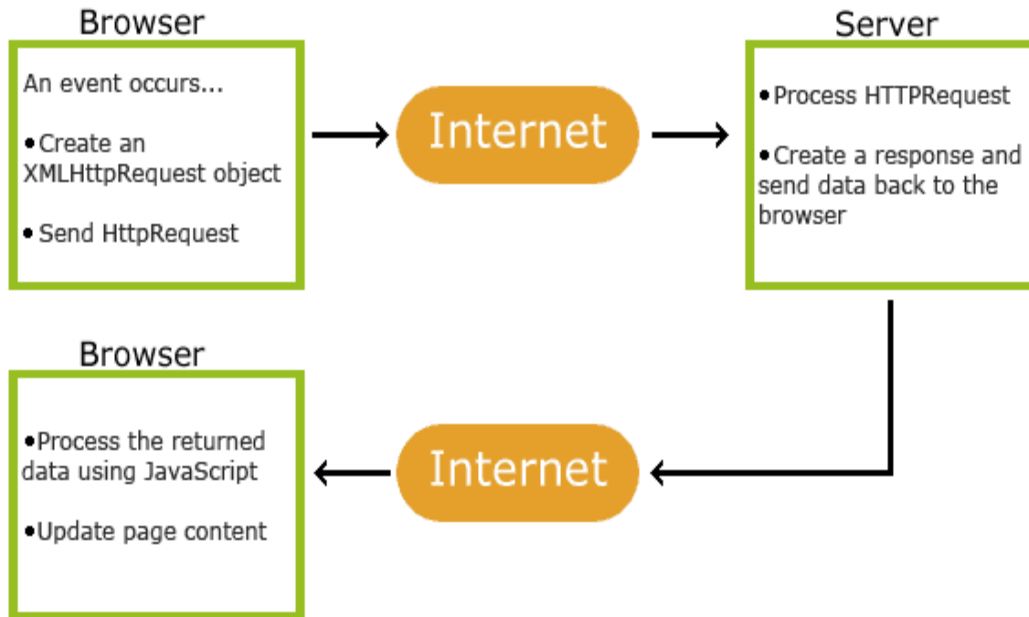


Figure 4. AJAX workflow

When an event occurs on a web page an XMLHttpRequest object is created by Javascript and sends a request to web server then the server processes the request and sends a response back to the web page, and proper action is performed by Javascript.

AJAX is very useful in the development because using this we can update a web page without reloading the page; request and receive data from a server and send data to a server in the background.

- **Python:** it is a widely used high-level programming language created by Guido van Rossum and first released in 1991. It uses whitespace indentation to delimit code blocks rather than curly brackets or keywords, it allows developers to code in fewer lines than using other languages like C++ or Java [13][14]. Python also provides automatic memory management and supports multiple programming paradigms

including object-oriented, imperative, functional and procedural styles. It has a large and comprehensive standard library [15].

When working complex biological data files it is very important to check the correctness of the files. Pandas is a software library written for Python and it is very efficient for file checks and for database connection at the backend. Pandas offer data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

- **MySQL:** We created Python functions to insert and retrieve data from required tables and we also created functions to automatically create a database for new organism's KB and we automatically create tables if the organism is already existing. We use phpMyAdmin which is a free software tool written in PHP and handles the administration of MySQL over the Web. phpMyAdmin supports many different MySQL operations with ease of use. It is an intuitive web interface and creates graphics of database layout in various formats.

Our main database is called "KBC_Admin" which has all tables belong to all organisms. It has two main tables which are used to store a library of tables – "Admin_user_common_db_library" and library of organisms – "organisms_library". These tables are used as references when retrieving data to show on the browser. Below I will show Entity–Relationship for important tables in the KBC_Admin database in two diagrams one for organisms' information and another for users' information.

Metadata – Organisms_library: Metadata is data about data, that is giving information about the data itself. Metadata representation about the organisms_library table would be

giving details about its attributes. In the database terminology, entity is logical representation of the data and table is its physical representation; attributes are its columns.

We can get information about attributes of a table in MySQL using this simple command:

“SHOW COLUMNS FROM <table>”

SHOW COLUMNS FROM organisms_library

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
database	varchar(500)	NO		NULL	
organism_id	int(11)	NO		NULL	
organism	varchar(500)	NO		NULL	
Genome_Version	varchar(500)	NO		NULL	
org_type_id	int(11)	NO		NULL	
organism_type	varchar(500)	NO		NULL	
model_organism	varchar(500)	NO		NULL	
org_logo_path	text	NO		NULL	

Figure 5. Organism library metadata

Entity-Relationship diagram for organisms: In this, we can see main tables including organisms_library which stores information about existing organisms including ID, version, name, type, logo, etc., which is connected to organism_list table which has list of all organisms with their unique ID, name, and type. This is a 1-1 relation because organism in organism library can only appear once as we create or already existing but exactly once. We can see the 1-many relation between organism_type, which stores 4 types of organisms, and organism_list; organism_type and organisms_library because many organisms can belong to the same type hence it is a 1-many relation. We also store user information in the Admin_users where have a user id, name, password, first_name, last_name, email, title, organization, etc., to keep track of user information and their organization. We also have

table `Admin_user_common_db_permission` which stores information about the owner of the data and group if they belong to any research group.

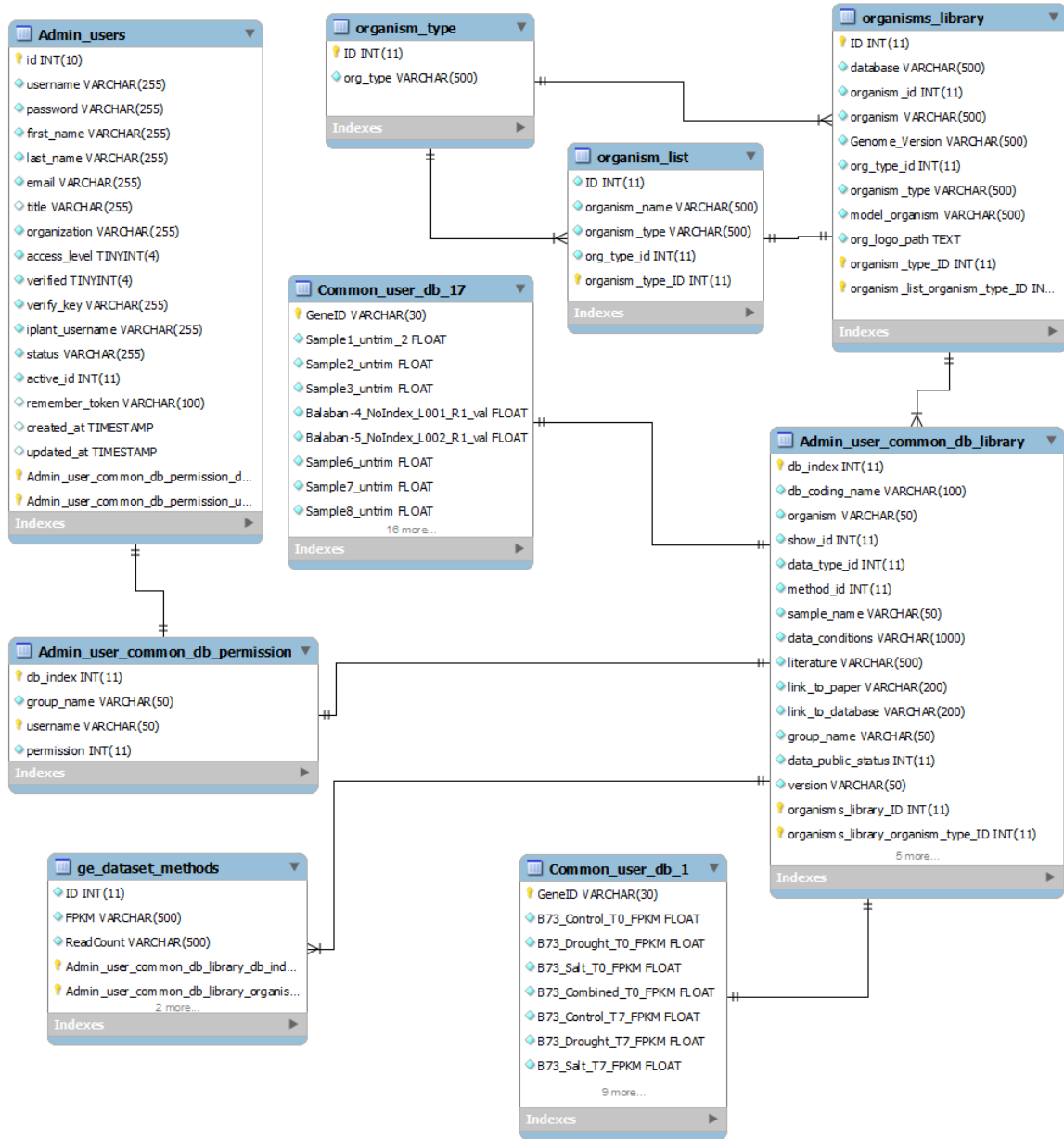


Figure 6. Entity-Relation diagram for organisms

The table `Admin_user_common_db_library` stores information about `common_user_db_1` to `common_user_db_n` which are created when user uploads data and they keep

information about what kind of data it is and which organism's data it is. It is like a library of tables which stores all table information. The common_user_db table has data for gene expression, differential expression, proteomic expression etc., It can be any data and is created automatically and information about this tables is stored in the library simultaneously. The number of common_user_db will be increasing as user uploads files hence in the below figure only two of them have been shown to make the diagram clear.

Metadata – Users Library:

SHOW FULL COLUMNS FROM Admin_user_common_db_library

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
db_index	int(11)	NULL	NO	PRI	NULL	auto_increment	select,insert,update,references	
db_coding_name	varchar(100)	latin1_swedish_ci	NO		NULL		select,insert,update,references	database coding name
organism	varchar(50)	latin1_swedish_ci	NO		NULL		select,insert,update,references	
show_id	int(11)	NULL	NO		0		select,insert,update,references	0 default, 1 FPKM pie chart, 2 DiffExp page
data_type_id	int(11)	NULL	NO		NULL		select,insert,update,references	1 FPKM, 2 TranscriptomicsRPKM RNA-Seq, 3 TranscriptomicsReadCount RNA-Seq, 4 Differential transcriptomics, 5 Transcriptomics Microarray Affymetrix, 6 Transcriptomics Microarray Agilent, 7 Proteomic MassSpectrometry GCMS, 8 Proteomic MassSpectrometry LCMS, 9 Proteomic 2Dgel, 10 Differential Proteomics, 11 Epigenomic Bisulphite, 12 Epigenomic Methylation Array, 13 Epigenomic MBD-Seq Array, 14 Differential Epigenomics
method_id	int(11)	NULL	NO		NULL		select,insert,update,references	1 cuffdiff, 2 edger, 3 VOOM, 4 Tophat, 5 limma, 6 Transcriptomic gene expression, 7 proteomic expression, 8 methylation data
sample_name	varchar(50)	latin1_swedish_ci	NO		NULL		select,insert,update,references	name
data_conditions	varchar(1000)	latin1_swedish_ci	NO		NULL		select,insert,update,references	conditions
literature	varchar(500)	latin1_swedish_ci	NO		NULL		select,insert,update,references	publications
link_to_paper	varchar(200)	latin1_swedish_ci	NO		NULL		select,insert,update,references	URL for publications
link_to_database	varchar(200)	latin1_swedish_ci	NO		NULL		select,insert,update,references	URL for database
group_name	varchar(50)	latin1_swedish_ci	NO		NULL		select,insert,update,references	
data_public_status	int(11)	NULL	NO		0		select,insert,update,references	0 private; 1 public
version	varchar(50)	latin1_swedish_ci	NO		NULL		select,insert,update,references	version of database

Figure 7. Metadata for user library

In this, we have data about admin_user_common_db_library table's columns. We can also see the comments about each columns in this display. Here we maintain data_type_id, method_id, and so on., to represent what type of data user has brought and how this is represented. We assigned id to each supported data type and method.

Entity-Relationship diagram for users:

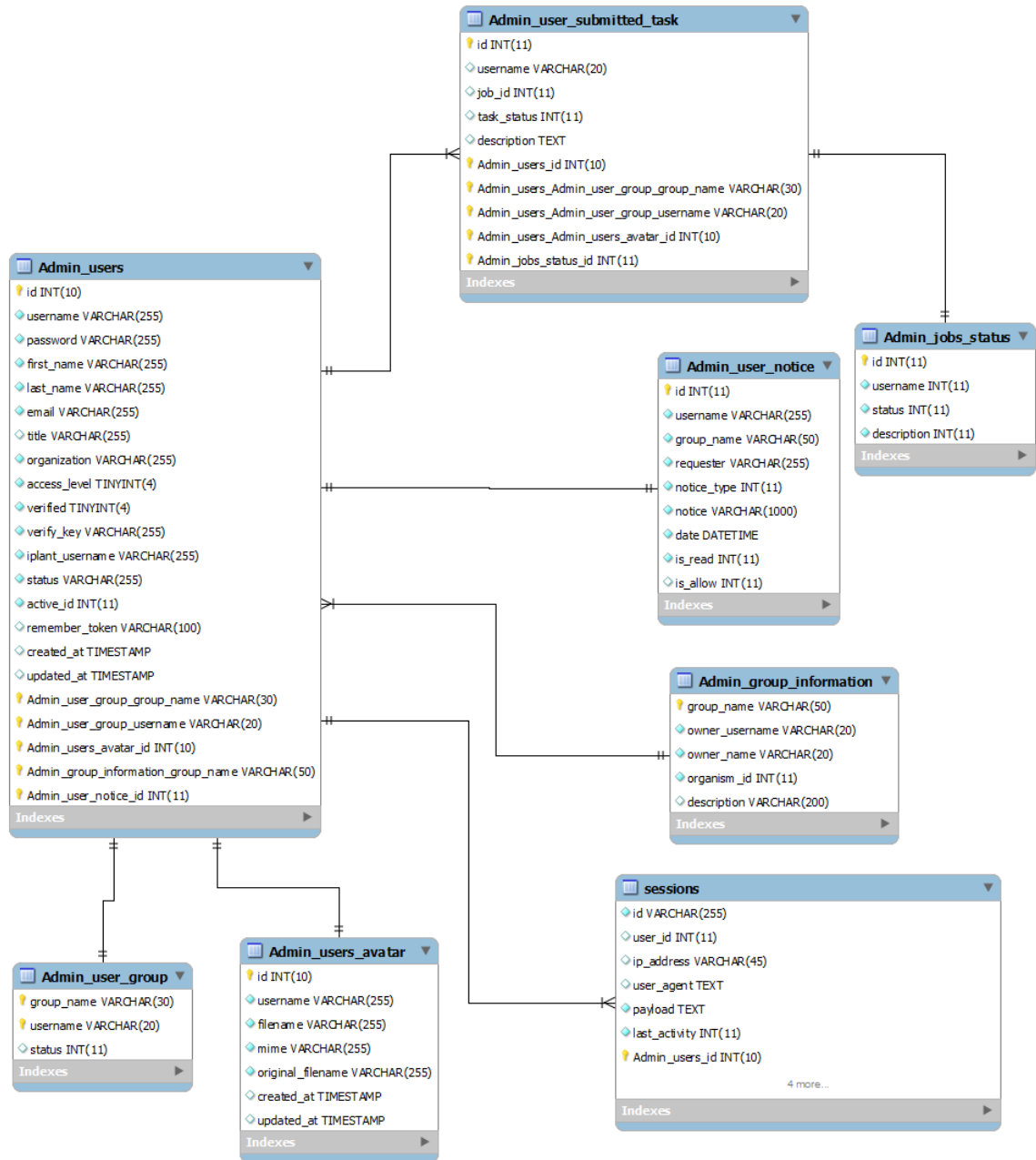


Figure 8. Entity-Relation diagram for users

User information is stored in the `Admin_users` table as explained above. I used `Admin_users` table in both ER diagrams to show the relation between users and organisms and their contribution of data to the organisms. Here we store group information if a user

belongs to any research group, this is very useful as we collaborate with many different groups than individuals and hence ownership of data can be given to the whole team. Admin_user_avatar stores users or group logo to display on their page this can be used to represent their main research area. We also have tables to store information about user sessions and tasks to keep track of who brings data and what kind of job they are doing and it can be used to retrieve session if they failed during any session.

- **GitHub:** it is a web-based version control repository and internet hosting service, mostly used for code. It provides all of the source code management and distributed version control functionalities of Git [16] and its own features. Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. Git can be used to keep track of changes in the set of files but mainly focused on source code management in software development [17].

It is very important to work with version control software when multiple people are working on the same project. In interdisciplinary and multidisciplinary fields like bioinformatics, no problem can be solved by experts from single domain hence collaboration with many people is required and the use of version control like Git and GitHub is very important and convenient to make the smooth flow of the work among multiple developers.

GitHub also offers both private and public repositories on the same account [18]. This is very convenient because we can keep projects private until the finish of the project and then can publish it as open-source software projects [19]. It is the largest host of source code in the world [20].

GitHub uses a few very basic Git commands for basic workflow as below:

- `git status` - displays what files have changed or been added or deleted.
- `git add -A` - adds all changes ("stages" the files). The A flag adds all changes.
- `git commit -m "your message"` - commits all changes and adds a message describing this change.
- `git push` - pushes your changes to the remote repository.

For synchronizing the changes – we can use ‘`git pull`’ which downloads all the changes from the repository and updates your local files. We can also use a combination of ‘`git fetch`’ which gets all the changes from the GitHub repository but not change local files; and ‘`git merge`’ this updates the local files with the changes obtained from fetch command.

- **Architecture Flexibility:** Another important feature of a good software development is to have place holders to incorporate any future developments as any project is ever evolving with new tools and functionalities. Our Laravel framework is very flexible to include any other future developments easily as we adopted agile software development life cycle for our project. Combination of Laravel architecture and Agile software cycle make our development adjustable to add new features and other components. For example, we could just modify routing options in the framework to include other web pages and functionalities.

3. KBCommons Functionalities

In this section, I will explain about the KBCommons procedure, supported data types and KBCommons features including Browse KBCommons, Contribute to KB, Add Version to KB, and Create a new KB.

3.1 KBCommons Flowchart

In KBCommons, first and basic thing that users can do is to browse existing organisms when they do not have any data about any organism and when they want to explore existing organisms with the existing data and perform data analysis on that data. The starting page is a home page which is the KBCommons main website: <http://kbcommons.org/>. When the user goes to this URL then the request (this is MVC cycle) will be sent to the **routes** as shown below:

```
Route::get('/system/home/kbc_system_main', 'System\KBCHomeController@KBCSystemMain')->name('system.home.kbc_system_main');
```

Figure 9. KBC home route

This request from routes will send information to the KBCHomeController and in that to the KBCSystemMain function, this is the **Controller** function and it is represented as 'System\KBCHomeController@KBCSystemMain' in the above figure. Here System is the folder where all controller functions for the main system are stored. The name 'system.home.kbc_system_main' is used as a reference to call this controller anywhere in the Laravel project. '/system/home/kbc_system_main/' is the URI that will be shown on the address bar like this: 'http://kbcommons.org/system/home/kbc_system_main'. We use descriptive names in the URI and in the reference names so that by looking at the URI we can understand at what stage of the procedure we are in.

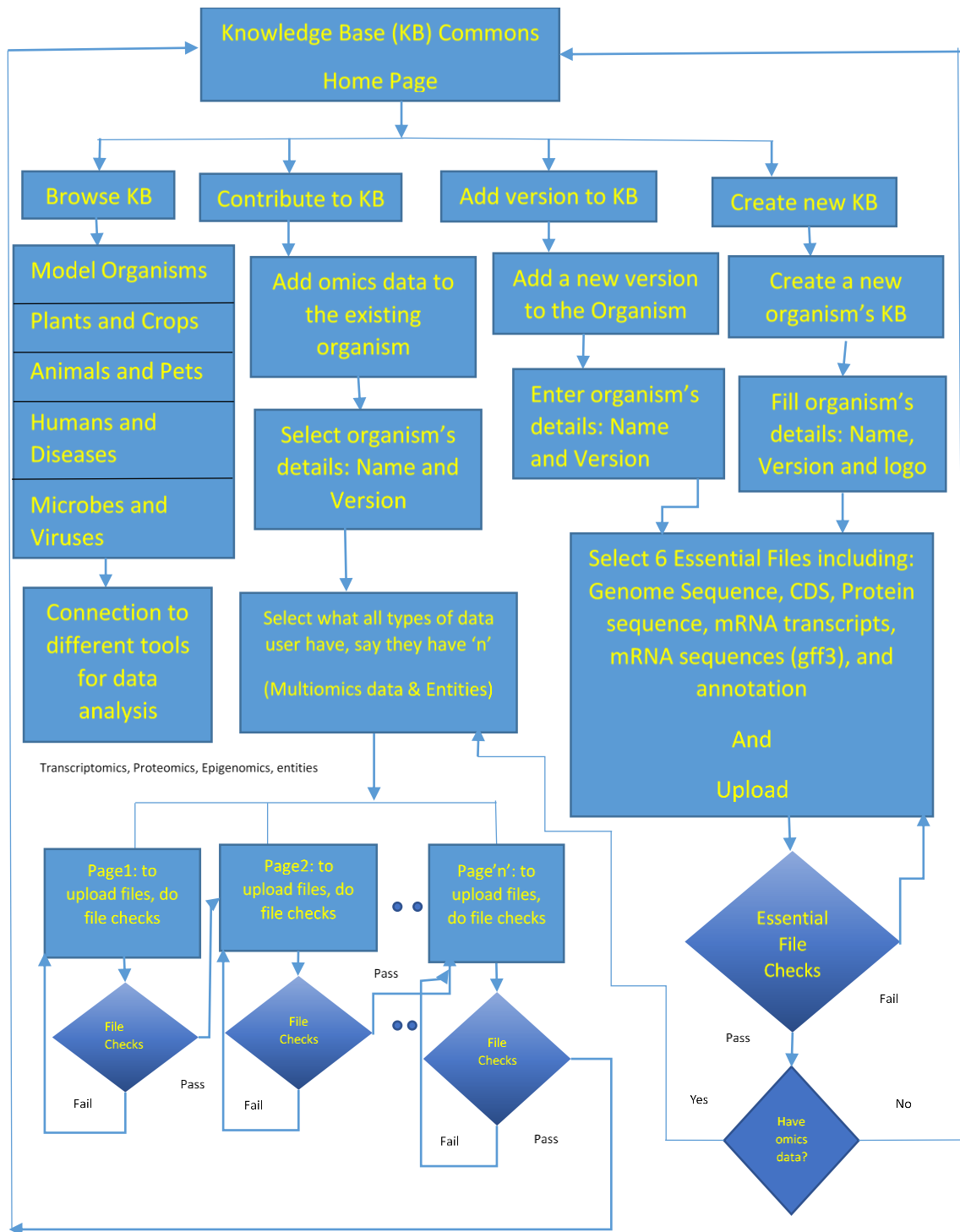


Figure 10. KBCommons Flowchart

From KBCCommons home page user can navigate to Browse KBCCommons, Contribute to KB, Add version to KB and Create a new KB. KBCCommons home page is shown below.

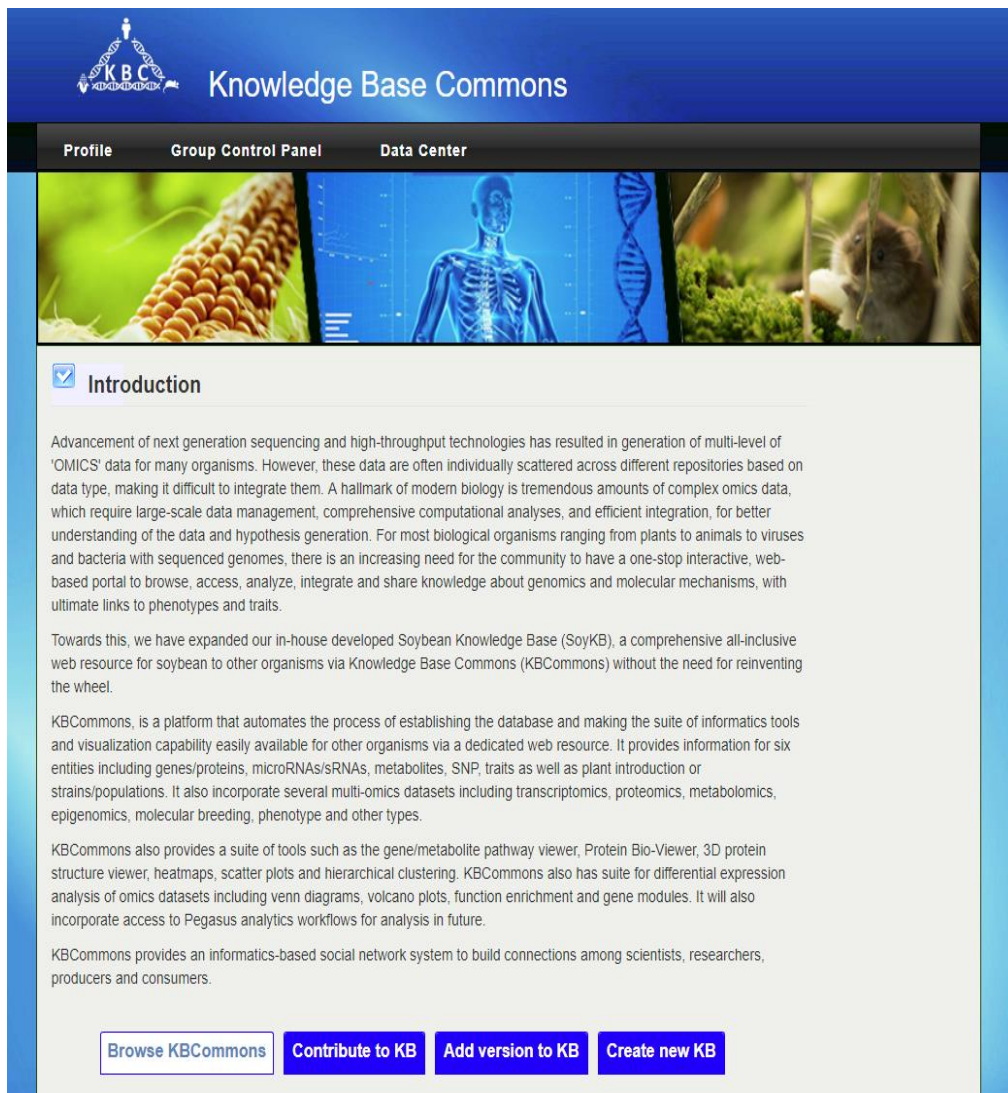


Figure 11. KBCCommons Home Page

In browse KB section, the user can explore already existing organisms' functionality and do data analysis using the existing data.

In Contribute to KB section, the user can upload omics or entities data for any existing organism and perform more complex functionalities based on the data.

In Add version to KB section, the user can add a new version to the organism if it is already existing.

And in Create a new KB section, the user can create a new organism's KB with basic essential files that are required to create a new KB.

Currently, in our KBCommons, we are supporting four main types including Plants and Crops; Animals and Pets; Humans and Diseases; Microbes and Viruses. This will facilitates researchers to find organisms belong to their interesting category and browse each section quickly. Study of organisms belong to Plants and Crops will help to increase the yield of field crops, and many of our early medicines come from plant extracts hence it will be useful in precision medicine as well. Animals and Pets organisms can also be used in human health research and can be helpful to protect pets from diseases as well. Studying organisms belong to Humans and Diseases is very useful and important in the field of precision medicine and for better and healthy living of humanity. Microbes and Viruses study will also be very useful as microbes have a profound impact on every facet of human life and everything around us, this can be finally helpful in precision medicine.

3.2 Data Sources and Supported Datatypes

In this section, I will explain about data types that we currently support in KBCommons and common resources where we could find those data.

First, to establish any organism's new KB in our KBCommons database we need 6 essential files which are mostly in FASTA format or txt format or gff3 file format. We can get these 6 essential files from Phytozome (<https://phytozome.jgi.doe.gov/>) which is widely used and well-cited resource for organisms belong to Plants and Crops [21].

Phytozome hosts 77 assembled and annotation genomes, from 74 viridiplantae species. It also supports different tools including BLAST (<https://blast.ncbi.nlm.nih.gov/Blast.cgi>), JBrowse (<https://jbrowse.org/>), PhytoMine (<https://phytozome.jgi.doe.gov/phytomine/>), and BioMart (<http://www.biomart.org/>).

Ensembl (<https://www.ensembl.org/>) is a very popular resource for organisms belong to Animals [22]. It is a genome browser for vertebrate genomes that supports research in comparative genomics, evolution, sequence variation and transcriptional regulation. Ensemble annotates genes, compute multiple alignments predicts regulatory function and collects disease data. It supports various tools including BLAST, BioMart, and the Variant Effect Predictor for all species it supports. It also creates and integrates datasets and analysis tools that enable genomics.

GenBank (<https://www.ncbi.nlm.nih.gov/genbank/>) is the National Institute of Health (NIH) genetic sequence database and it has annotated collection of all publicly available DNA sequences [23]. GenBank exchanges data with the DNA DataBank of Japan (DDBJ) [24][25] and the European Nucleotide Archive (ENA) [26] on a daily basis which ensures worldwide data coverage. GenBank is accessible through the NCBI Entrez retrieval system, which integrates data from the major DNA and protein sequence databases along with taxonomy, genome, mapping, protein structure and domain information, and the biomedical journal literature via PubMed. The GenBank database is designed to provide and encourage access within the scientific community to the most up to date and comprehensive DNA sequence information.

These are the main common data resources we could get data to create a new organism's KB or to upload multi-omics data and entities information to already existing organism.

Table 1. Data types for multi-omics and entities

Category	Type	Dataset	Datatype	Dataset Type	Dataset Method	
Multi-Omics Data	Transcriptomics	Gene Expression	RNA-Seq	ReadCount	HTSeq	
				FPKM/RPKM	Samtools	
			Microarray	Affymetrix	Cufflinks	
					Tuxedo	
					RMA	
			Agilent	Lowees		
		Differential Expression Genes				EdgeR
						Limma
						Voom
						Cuffdiff
	Proteomics	Proteomic Expression		Mass Spectrometry	GCMS	
				2DGel	LCMS	
		Differential Expression Proteins				
		Epigenomics	Methylation	Bisulphite Sequencing		CG
					CHG	
					CHH	
	Methylation Array				CG	
					CHG	
				CHH		
MBD-Seq Array			CG			
			CHG			
		CHH				
Differential Methylated Regions						
Entities	miRNA/sRNA	Expression				
		Target gene				
		Differential Expression miRNA				
	SNP – Inserted/deleted	SNP Array Types				
		GWAS				
		SNP dinucleotide				
	Metabolite					
	Plant Introduction Lines/ Animal					
	Phenotype/ TRAIT					

These are the supported data types for uploading multi-omics data and entities.

3.3 Browse KBCommons

Here, we display all the existing organisms with their versions. We categorized organisms into four main types including Plants and Crops; Animals and Pets; Humans and

Diseases; Microbes and Viruses. Along with this classification, we also have Model Organisms section when we display model organisms from all the categories.

In our current KBCommons under Plants and Crops, we have *Arabidopsis thaliana* and *Zea mays*; under Animals and Pets, we have *Mus musculus*; under Humans and Diseases, we have *Homo Sapiens*. Among these *Arabidopsis thaliana* and *Mus musculus* are considered as Model Organisms. We save the different type of organisms in corresponding arrays and we call that view with this information.

```
<?php
namespace App\Http\Controllers\System;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
//use Illuminate\Support\Facades\Validator;
use App\KBClasses\Home\DBAdminHomeDatabaseWrapperClass;
use Session;
use File;

class KBHomeController extends Controller
{
    //

    public function KBCSystemMain(){

        $db_admin_home_db_wrapper = new DBAdminHomeDatabaseWrapperClass();
        $data = $db_admin_home_db_wrapper->fetchAllOrganismObjArray();

        $mod_organism_array = array();
        $pac_organism_array = array();
        $aap_organism_array = array();
        $had_organism_array = array();
    }
}
```

Figure 12. KBCHome controller system-1

For these the code snippet of the called controller function is shown in Figures 12 and 13: PHP and MySQL are used to retrieve data and send that to the view. Like this, all existing organisms' information is retrieved from the corresponding table and stored in corresponding arrays. For each existing organism, all the information including logo,

existing genome, and model versions are stored in the 'info' array and sent to the view to display on the website.

```
        $mod_org_logo[] = $org_logo;
        $mod_org_gm[] = $org_gm_array;
    }
}

$info = [
    "mod_organism_array" => $mod_organism_array,
    "pac_organism_array" => $pac_organism_array,
    "aap_organism_array" => $aap_organism_array,
    "had_organism_array" => $had_organism_array,
    "mod_org_logo" => $mod_org_logo,
    "pac_org_logo" => $pac_org_logo,
    "aap_org_logo" => $aap_org_logo,
    "had_org_logo" => $had_org_logo,
    "mod_org_gm" => $mod_org_gm,
    "pac_org_gm" => $pac_org_gm,
    "aap_org_gm" => $aap_org_gm,
    "had_org_gm" => $had_org_gm,
];

return view('system/KBCSystemMain')->with("info", $info);
}
```

Figure 13. KBCHome controller system-2

This view is the KBCSystemMain.blade.php which is stored in system folder which is under views folder as shown in the above figure. Laravel is very efficient that we do not have to give absolute path because all default paths of views, controllers, routes, etc are already configured. Code snippet of KBCSystemMain.blade.php: where we write HTML code along with some PHP and Javascript to perform functionality and to display the data in the browser.

```

<Form id="autoformatted_form" action="{{ route('system.home.kbc_home_router') }}" method="get"
>
  <div class="hor-bar">
  <div class="heading-bar"><h3>Model Organisms</h3></div>
  <br>
  @foreach ($mod_organism_array as $index => $value)
    <div class="img-with-text">
      <div class="img-logo"></div>

      <div class="txt-logo"><h5>
        <select id="version" name="version" class="inputs">
          <option>version</option>
          @foreach ($mod_org_gm[$index] as $version)
            <option value="{{ $version }}">{{ $version }}</option>;
          @endforeach
        </select>
        <input type="submit" value="{{ $value }}" name="organism"> </h5>
      </div>
    </div>
  @endforeach
</div>
<br><br>
<div class="hor-bar">
<div class="heading-bar"><h3>Plants and Crops</h3></div>
<br>

```


Figure 14. KBCSystemMain view

From the above code, we can see that we are using nested for loop to loop through each organism in each category and display logo and existing versions. When user clicks on any of the existing organisms it will be redirected to that particular organism’s page, this is done by calling the action ‘system.home.kbc_home_router’, this is the name of the controller function ‘KBCHomeRoute’ in KBCHomeController from the routes. Here we send organism name information to the controller function to display the corresponding data of that selected organism. We store organism logo information in one array and existing versions information in a different array and display existing versions as a dropdown list.


On that particular organism page, the user can browse existing data using various tools including Blast, Multiple Sequence Alignment, Motif Sampler, Phylogeny, etc., by going to that particular page. Users can also visualize gene expression and differential expression data via pie charts and plain text.

[Browse KBCommons](#)
[Contribute to KB](#)
[Add version to KB](#)
[Create new KB](#)

Model Organisms




version ▼ Athaliana




version ▼ MusMusculus

Plants and Crops




version ▼ Athaliana




version ▼ Zmays

Animals and Pets



version ▼ MusMusculus

Humans and Diseases



version ▼ homoSapiens

Figure 15. Browse KBCommons organisms

For each organism, we show existing versions in the format GenomeVersion_ModelVersion to give the user an overview of all the existing organisms along with their existing versions. For example for *Zmays*: we have AGPv2_5b and AGPv3_6a where AGPv2 is genome version and 5b is its model version; AGPv3 is genome version and 6a is its model version as shown below:

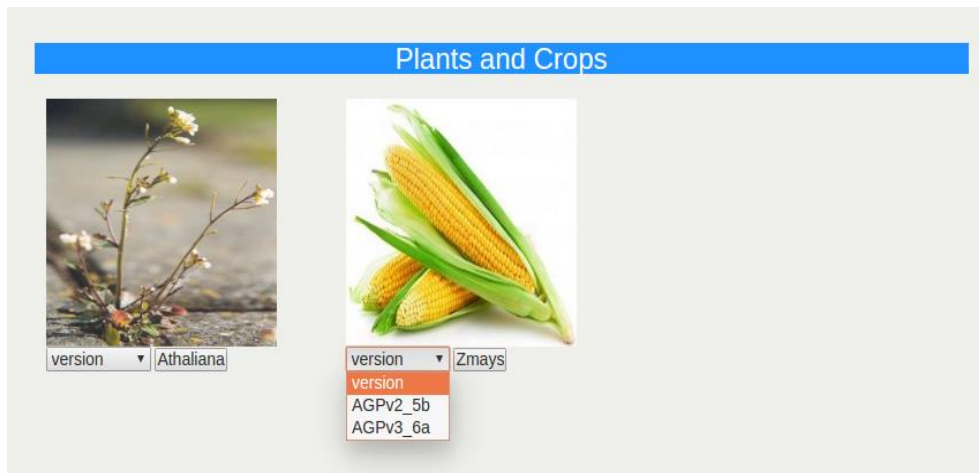


Figure 16. Displaying organism's versions

With this explicit display of existing versions of each organism user could decide by looking at the list if user has data for a new version and wants to perform data analysis for this new version they could go to Add version to KB page to do so or if they want to browse this existing version they could click on that particular organism and explore existing data.

In organism's home page user can search genes, browse differential expression data and visualize comparisons of different conditions and see data in different tools including blast, pathway viewer etc., for data analysis based on the availability of the existing data for that particular organism.

3.4 Contribute to KB

After exploring existing data of an organism if the user wants to perform data analysis with different data then the user can upload new data to that particular organism. Here, the user can select multi-omics data and also different entities.

The first step in this is to select organism's details including what type of organism that means whether it belongs to Plants or Animals or Diseases or Viruses. Based on that in the dependent dropdown list of existing organisms corresponding to that type will be displayed and based on that selected organism existing list of versions will be displayed for the user to select.

The user must select all details including type, name, and version else error message will be displayed. Upon successful selection of details and clicking submit button user will be redirected to a page where the user can select what types of data they have. This page gives an overall idea to the user what all the data types we are supporting. Currently, supported datatypes are given in the Table1 in Data Sources section 3.2.

User has to select at least one dataset for any data type else error message will be displayed. Upon successful selection, user will be redirected to a particular page based on the selection. User can select both gene/protein/methylation data and differential expression data for any data type and can have many datasets for each gene/protein/methylation data and for each data set they have to provide details including dataset type, dataset privacy that is whether this data is private or can it be public, name and method of the dataset. We also need Number of conditions and number of replicates for each dataset selected and based on that we restrict on a maximum number of differential expression files that they can upload. For example, if they have 'c' number of conditions

then the maximum number of comparisons can be $c*(c-1)/2$. Hence a maximum number of differential expression file can be $c*(c-1)/2$.

Apart from a number of file checks we also perform extensive checks on each selected data set for the correctness of the files and then only we upload them to the database. A few important file checks are mentioned below:

1. Compatibility with the number of columns in the file and the number of conditions provided

2. Emptiness is not allowed in certain data types

3. Duplicate values are not allowed

4. Gene names in the files must be consistent with the gene names in the genome file of the organism and selected version

5. For differential expression file, we also deal with $-\text{inf}$ and inf value by recalculating the log fold change value

6. There are other checks based on the specific data type selected, for example, if user selected ReadCount data type for RNA-Seq gene expression then we make sure that file has only integer values

Like these, we do extensive file checks for each data set user wants to upload based on the data type selected. Supported data types for multi-omics and entities are shown in Table1. After successful file checks and uploading to the database this data will be available to perform data analysis and to display on the organism's home page.

I will explain all details and conditions for each data type using a sample organism as a case study at the end of this chapter.

3.5 Add Version to KB

If an organism is existing and the user wants to add a new version to the existing organism then the user can do it here.

It mainly involves three steps, two mandatory and one additional step including:

1. Fill the organism details

- i. In this, user will add organism type, name, model version and genome version
- ii. Upon selecting organism type we display existing organism names for selecting and then user will enter new version

2. Upload 6 essential files

- i. User must bring 6 essential files of that organism's version to be able to add new version
- ii. The basic files to establish a new version including
 1. Nucleotide FASTA format of the Genome Sequence File
 2. Nucleotide FASTA format file of all gene coding sequences
 3. Amino acid FASTA format file of all gene coding sequences
 4. Nucleotide FASTA format file of spliced mRNA transcripts
 5. GFF3 format representation of all mRNA sequences
 6. A summary of function annotation details in txt format
- iii. We upload files only after all 6 required files are selected with proper file formats
- iv. After this user have an option to go to home page to browse this version or if they have multi-omics data then they can continue uploading files

3. Upload Multi-omics data

- i. *Refer to 2.3 Contribute to KB section, same as that*

3.6 Create a new KB

Creating a new KB for an organism can be done when there is none existing already. This is similar to add new version except that here the user has to select new organism details along with organism logo to indicate about the organism.

It has three main steps, two mandatory and one additional step including:

1. Fill the organism details

- i. In this, user will add organism type, name, model version, genome version and upload organism logo – a small image file that should describe organism
- ii. Upon selecting organism type we display new organism names for selecting and then user will enter new version and upload logo

2. Upload 6 essential files

- i. *Refer to 2nd point upload 6 essential files in 2.2 Add version to KB section, same as that*

3. Upload Multi-omics data

- i. *Refer to upload multi omics in 2.3 Contribute to KB section, same as that*

4. Case Study

In this section, I will explain the working flow of KBCommons from both biological and technical perspective with *Medicago truncatula* organism as an example.

4.1 *Medicago truncatula* Introduction

Medicago truncatula is studied as a model organism for legume biology because it has a small diploid genome, is self-fertile, has a rapid generation time and prolific seed production, is amenable to genetic transformation, and its genome has been sequenced [27]. It is native to much of southern and eastern Europe, as well as northern Africa, Western Asia and the Caucasus. It is also naturalized in Australia and parts of South America. Its common name is Barrel medic and it is adapted to different soil textured types, sandy to clayey, but particularly well drained neutral to alkaline soils, pH 6 to 8.

A model organism is a non-human species that is extensively studied to understand particular biological phenomena, with the expectation that discoveries made in the organism model will provide insight into the workings of other organisms [28]. Model organisms are in vivo models and are widely used to research human disease when human experimentation would be unfeasible or unethical [293]. This strategy is made possible by the common descent of all living organisms, and the conservation of metabolic and developmental pathways and genetic material over the course of evolution [30].

Medicago truncatula can be used for a gene donor for crop improvement. It is used as animal forage and fodder, for soil improvement and the prevention of soil erosion and as a companion crop [31].

Definitions of model organism and figure of *Medicago truncatula* is collected from Wikipedia (https://en.wikipedia.org/wiki/Medicago_truncatula).



Figure 17. *Medicago truncatula* A17 showing the shoot with leaves and seed pods

In this example, we are studying *Medicago truncatula* which belong to Plants and Crops type and it is also a model organism for legume biology so it will also be shown under model organism category. The organism under this study has genome version of Mt4 with model version V1.

4.2 Create a new KB for *Medicago truncatula*

If any organism which is not shown under browse organisms on the home page of KBCommons then the user can create a new KB for that organism. In this case, *Medicago truncatula* is not already existing in our KBCommons so we will create a new KB for this.

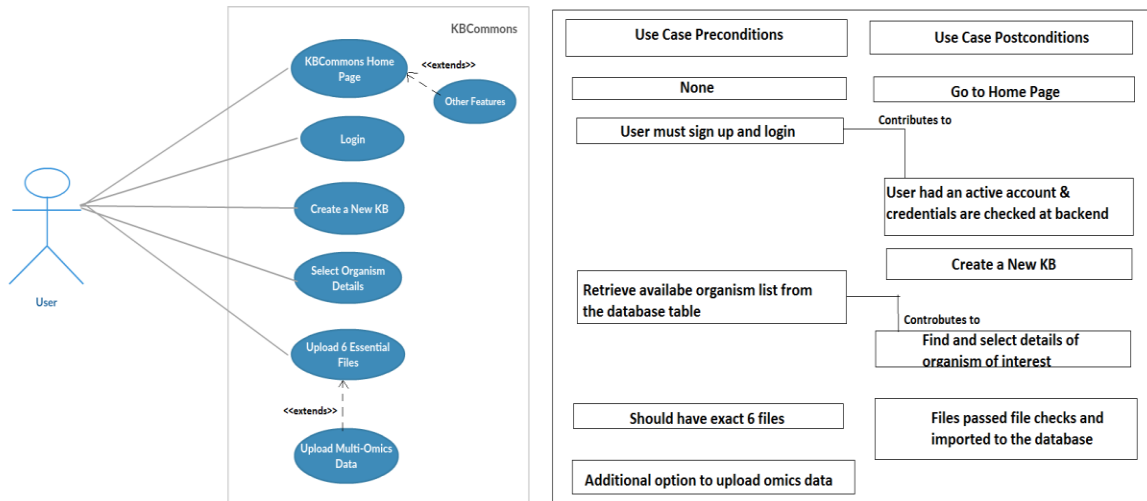


Figure 18. Create a newKB Usecase diagram

All steps for creating a new KB will be shown with data for *Medicago truncatula* with screenshots for each step. In this section, I will show only with 6 essential files and in Contribute to KB section for this same established KB, we will upload omics data files.

From KBCommons home page (<http://kbcommons.org/>) user have to click on Create a New Organism as shown below.

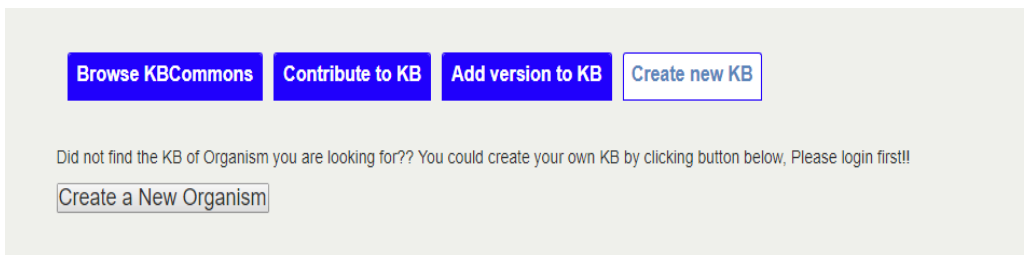
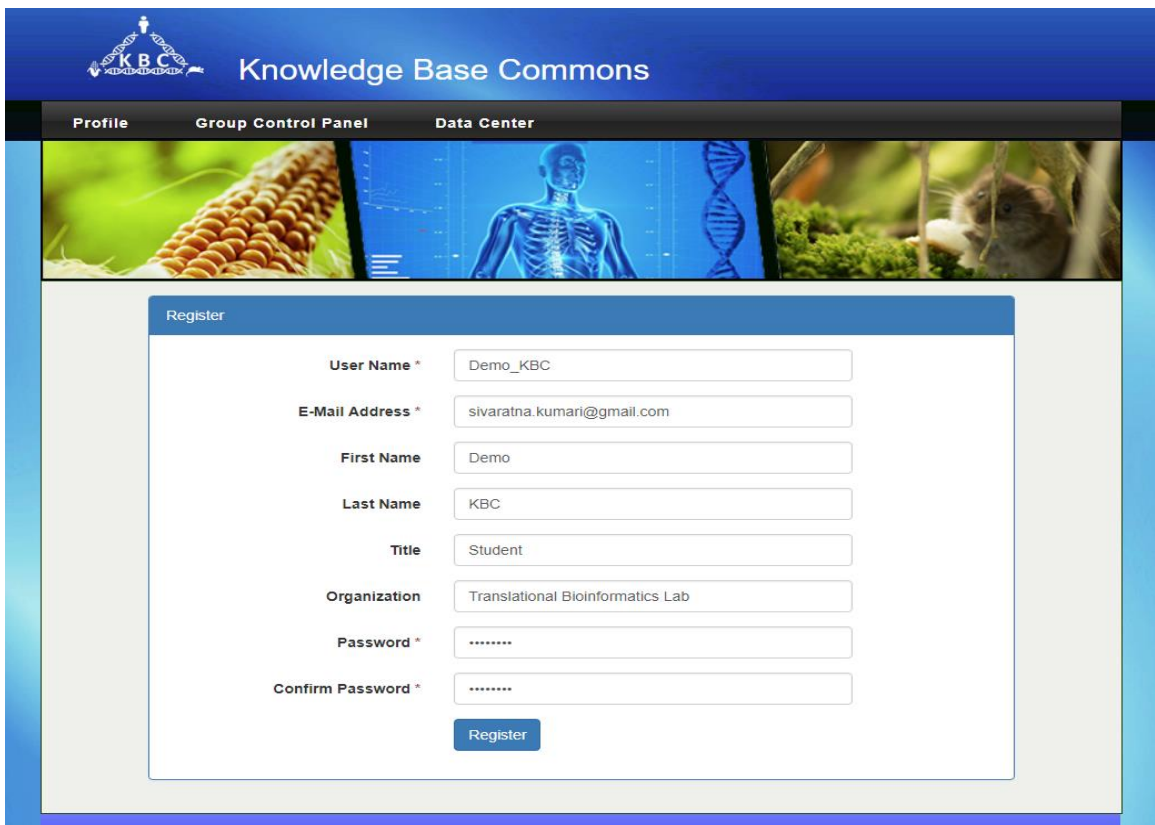


Figure 19. Create a new organism

When click on create a new organism, the request will be sent to the routes and to the corresponding controller function. We will send organism list for which users can create a new KB to the view to display on the browser.

The user must log in to create a new organism page because we keep track of owner information who uploads the data. If the user does not have a login then the user can sign up for a new account and then log in. To sign up user must provide us with User Name, E-Mail Address, and a password. We also ask the user to provide First Name, Last Name, Title, and Organization. This information would be useful later when dealing with groups and permissions. Especially Organization information would be useful to keep track of the type of users using KBCCommons and we could research and expand to other domain of users as well. The title could be the designation of the user for example if the user is a student or faculty or employee or researcher etc., Sample sign up and log in for a Demo user is shown in the below screenshots.



The screenshot shows the KBC Commons website interface. At the top, there is a blue header with the KBC Commons logo and the text "Knowledge Base Commons". Below the header, there is a navigation bar with three tabs: "Profile", "Group Control Panel", and "Data Center". The main content area features a banner image with a corn cob, a human skeleton, a DNA double helix, and a mouse. Below the banner is a "Register" form with the following fields:

Register	
User Name *	<input type="text" value="Demo_KBC"/>
E-Mail Address *	<input type="text" value="sivaratna.kumari@gmail.com"/>
First Name	<input type="text" value="Demo"/>
Last Name	<input type="text" value="KBC"/>
Title	<input type="text" value="Student"/>
Organization	<input type="text" value="Translational Bioinformatics Lab"/>
Password *	<input type="password" value="....."/>
Confirm Password *	<input type="password" value="....."/>
<input type="button" value="Register"/>	

Figure 20. KBC user sign up

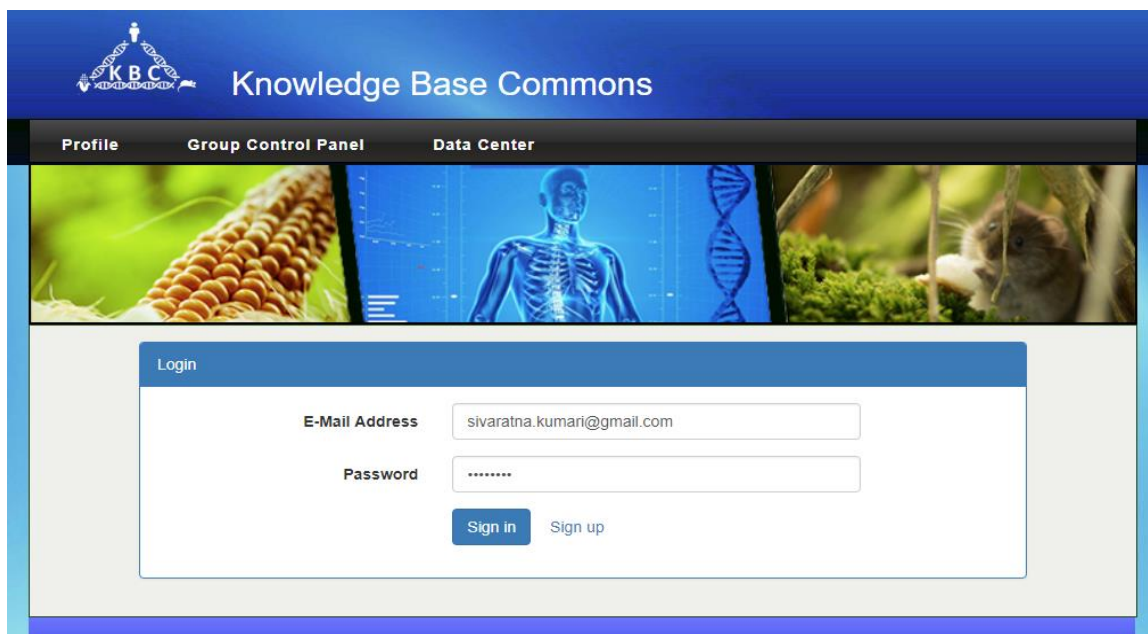


Figure 21. KBC user log in

After successful registration user can log in to be able to access all KBCCommons functionalities. After login and clicked on create a new KB the first thing to do is to provide the organism's details.

Step1: Fill the Organism's Details: After successful signup-login user can go to create a new organism's KB and select details about the organism as the first step. In this we have a dependent drop-down list of organisms; this list is collected from different websites for a different type of organisms and we update the list frequently to keep the list updated with the new organisms. Providing users with the list of organisms for creating KB also solves the problem of naming conflicts. If we allow users to type the name of the organism then one user may write *Medicago truncatula* and another user may type *Mtruncatula* hence to deal with this issue we provided users with a dynamic dropdown list of organisms.

```

$(document).on('change', '.organismtype', function() {
    var type_id=$(this).val();
    var op="";
    $.ajax({
        type: 'get',
        url: '!!URL::to('/dataCenter/newkb/kbc_findOrganismName_exist')!!)',
        data: {'id':type_id},
        success: function(data) {
            op+ '<option value="0" selected disabled> Select organism name</option>';
            for(var i=0; i<data.length;i++){
                op+ '<option value="'+data[i].ID+'">'+data[i].organism+'</option>';
            }
            $('#organismname').html(" ");
            $('#organismname').append(op);
        },
        error: function() {
        }
    });
});

```

Figure 22. Javascript dynamic dropdown

This dynamic dependent dropdown is created using javascript. First when the user selects the type of organism then a list of organisms belong to that type will be displayed on the same page. Based on the selection request will be sent to the controller function which is KBCFindOrganismName in KBCnewKBCController from a javascript function. This controller is named as '/dataCenter/newkb/kbc_findOrganismName' in the routes hence calling with this name from a javascript function. This list does not contain existing organisms. Here we have three functions one to retrieve a list of existing organisms, other to retrieve a list of other organisms and another one to retrieve a list of existing organisms' version details.

```

Route::get('/dataCenter/newkb/kbc_findOrganismName', 'DataCenter\newKB\KBCnewKBCController@KBCFindOrganismName')->
name('datacenter.newkb.kbc_findorganismname');

Route::get('/dataCenter/newkb/kbc_findOrganismName_exist', 'DataCenter\newKB\KBCnewKBCController@KBCFindOrganismNameExist')->
name('datacenter.newkb.kbc_findorganismname_exist');

Route::get('/dataCenter/newkb/kbc_findOrganismVersion', 'DataCenter\newKB\KBCnewKBCController@KBCFindOrganismVersion')->
name('datacenter.newkb.kbc_findorganismversion');

```

Figure 23. KBC find organism list

Medicago truncatula is a model organism and it belongs to plant and crops category with Mt4.0 genome version and v1 model version [32]. The data is collected from Phytozome version 12.1 (<https://phytozome.jgi.doe.gov/>). Phytozome is the Plant Comparative Genomics portal of the Department of Energy's Joint Genome Institute (JGI), that provides JGI users and the broader plant science community a hub for accessing, visualizing, and analyzing JGI-sequenced plant genomes, as well as selected genomes and datasets that have been sequenced elsewhere. As of release v12.1, Phytozome has 77 assembled and annotation genomes, from 74 viridiplantae species. 43 of these genomes have been sequenced, assembled and annotated with JGI Plant Science program resources.

Knowledge Base Commons siva ▾

Profile Group Control Panel Data Center

Create a New Organism's KB

Step 1 Fill the Organism Details → **Step 2** Upload 6 Essential Files → **Step 3** Upload Multiomics Data

Organism Type: Model Organism:

Organism Name: Existing KB organisms are not included here, please visit [Home](#) to browse

Genome Version:

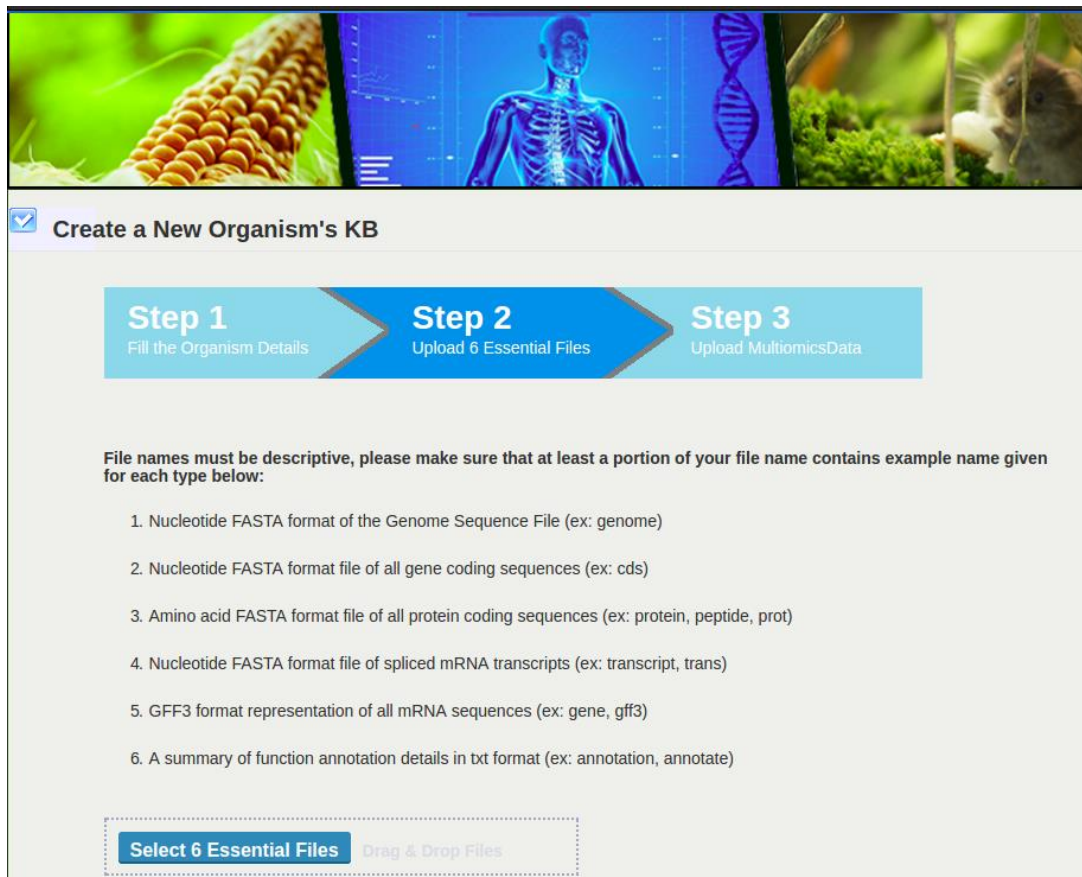
Model Version:

Organism Logo: 220px-Colora...lumbine.jpg

Figure 24. Create a new KB for *Mtruncatula*

Step2: Upload 6 Essential Files: After successfully submitting the details of the organism then step2 Upload 6 Essential files will be displayed.

In this user has to select 6 essential files with correct formatting and with descriptive names. For example whole genome sequence file should have 'genome' in the name, gene coding sequence should have 'cds' in the file name, amino acid coding sequence should have 'protein' or 'peptide', mRNA transcripts file should have 'cdna' in its file name, and annotation file is expected to have 'annot' or 'annotation' in the file name. This will help us to differential among file types to store data in the database and to retrieve to display on the browser.



Create a New Organism's KB

Step 1
Fill the Organism Details

Step 2
Upload 6 Essential Files

Step 3
Upload MultiomicsData

File names must be descriptive, please make sure that at least a portion of your file name contains example name given for each type below:

1. Nucleotide FASTA format of the Genome Sequence File (ex: genome)
2. Nucleotide FASTA format file of all gene coding sequences (ex: cds)
3. Amino acid FASTA format file of all protein coding sequences (ex: protein, peptide, prot)
4. Nucleotide FASTA format file of spliced mRNA transcripts (ex: transcript, trans)
5. GFF3 format representation of all mRNA sequences (ex: gene, gff3)
6. A summary of function annotation details in txt format (ex: annotation, annotate)

Select 6 Essential Files Drag & Drop Files

Figure 25. Select 6 Essential Files



Figure 26. Less number of essential files

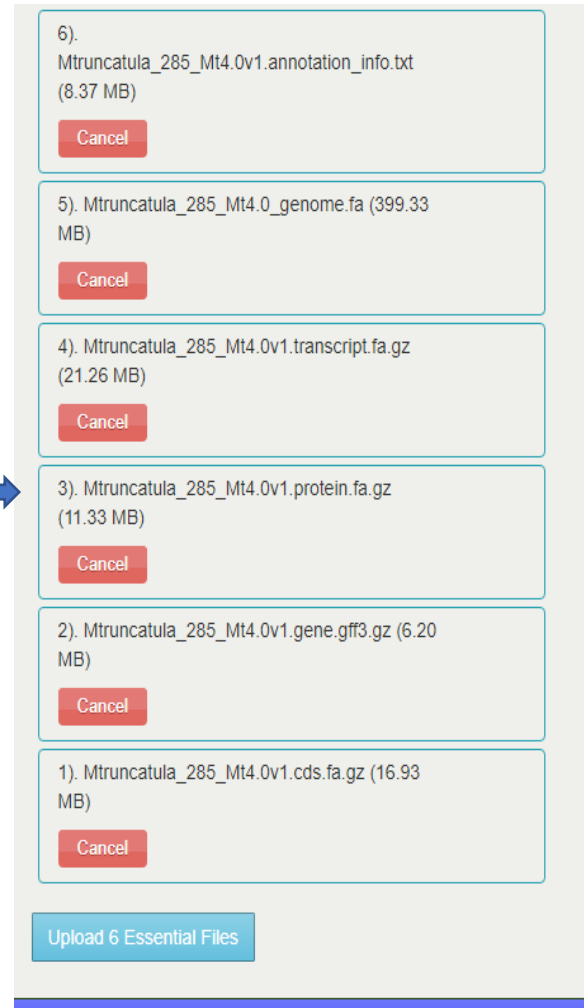


Figure 27. Correct 6 essential files

This is a case when the user selected less number of essential files hence upload button is not visible. We also check for duplicate files, file extensions, and there must be exact 6 number of files but not more or less. We also allow users to upload compressed files because these files can be very large and they may take a long time to upload if not compressed. Only after all conditions are satisfied then only upload files will be appeared. All these file extensions and naming conventions are decided by collecting data from main data resources for these organisms like from Phytozome and Ensembl.

After correct 6 essential files are selected we display size and names of each file to facilitate user to check correctness files again; if found incorrect files then the user can cancel any of them and select the appropriate files again and then upload the selected files.

6). Mtruncatula_285_Mt4.0v1.annotation_info.txt (8.37 MB)

5). Mtruncatula_285_Mt4.0v1.genome.fa (399.33 MB)

4). Mtruncatula_285_Mt4.0v1.transcript.fa.gz (21.26 MB)

3). Mtruncatula_285_Mt4.0v1.protein.fa.gz (11.33 MB)

2). Mtruncatula_285_Mt4.0v1.gene.gff3.gz (6.20 MB)

1). Mtruncatula_285_Mt4.0v1.cds.fa.gz (16.93 MB)

Upload 6 Essential Files

Figure 28. Upload 6 essential files

6). Mtruncatula_285_Mt4.0v1.annotation_info.txt (8.37 MB)

5). Mtruncatula_285_Mt4.0v1.genome.fa (399.33 MB)

4). Mtruncatula_285_Mt4.0v1.transcript.fa.gz (21.26 MB)

3). Mtruncatula_285_Mt4.0v1.protein.fa.gz (11.33 MB)

2). Mtruncatula_285_Mt4.0v1.gene.gff3.gz (6.20 MB)

1). Mtruncatula_285_Mt4.0v1.cds.fa.gz (16.93 MB)

Import to Database

Figure 29. Import 6 files to DB

While uploading 6 essential files we show the status of file uploading. It may take some time to upload if the file size is big. After the files are successfully uploaded on local storage then the user can import them to the database to make the data available on the browser for retrieval.

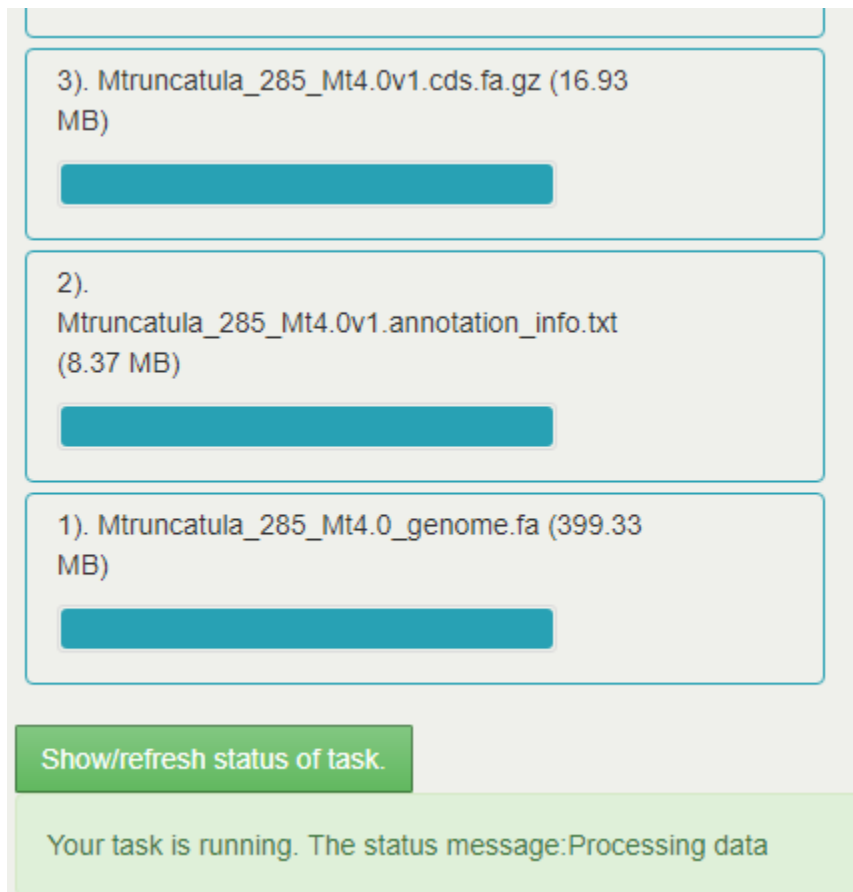


Figure 30. Import 6 files to Database Status

We display the status of importing files to the database and after files are successfully imported into the database then the next step is to upload Multi-omics data which is an optional step. By now our organism's new KB has been created with the uploaded 6 essential files. The user can choose to upload the multi-omics file if they have or the user can browse this organism further by going to this organism's page.

This organism has been added to the existing organisms' list and will display on our main page for browsing and also added in the backend database. (also display that option to go to home or to do multi-omics file uploads)

```

SELECT *
FROM `organisms_library`
LIMIT 0, 30

```

ID	database	organism_id	organism	Genome_Version	org_type_id	organism_type	model_organism	org_logo_path
1	KBC_Athaliana	6	Athaliana	TAIR10	1	Plants and Crops	on	http://artemis.cyverse.org/siva/test/KBCCommons_mul...
2	KBC_HomoSapiens	147	homoSapiens		3	Humans and Diseases		http://artemis.cyverse.org/siva/test/KBCCommons_mul...
3	KBC_MusMusculus	65	MusMusculus	GRCm38_83	2	Animals and Pets	on	http://artemis.cyverse.org/siva/test/KBCCommons_mul...
4	KBC_Zmays	64	Zmays	AGPv2_GM_5b.AGPv3_GM_6a	1	Plants and Crops		http://artemis.cyverse.org/siva/test/KBCCommons_mul...
5	KBC_Alyrata	5	Alyrata	1_1	1	Plants and Crops		http://artemis.cyverse.org/siva/test/KBCCommons_mul...
6	KBC_Mtruncatula	38	Mtruncatula	Mt4_0v1	1	Plants and Crops	on	http://artemis.cyverse.org/siva/test/KBCCommons_mul...

Figure 31. Organisms Library table

This organisms_library table stores all information about the organism including the database name, organism unique id, organism name, version, organism category, and path. KBC_Mtruncatula is the database where all the uploaded files for *Mtruncatula* will be imported to and retrieved from.

Table	Action	Rows	Type	Collation	Size	Over
Mtruncatula_cDNASequence_Mt4_0v1	Browse Structure Search Insert Empty Drop	~64,272	InnoDB	latin1_swedish_ci	112.6 MiB	
Mtruncatula_CDSSequence_Mt4_0v1	Browse Structure Search Insert Empty Drop	~58,462	InnoDB	latin1_swedish_ci	86.6 MiB	
Mtruncatula_ChromosomalPosition_Mt4_0v1	Browse Structure Search Insert Empty Drop	~419,601	InnoDB	latin1_swedish_ci	43.6 MiB	
Mtruncatula_KOG_Mt4_0v1	Browse Structure Search Insert Empty Drop	~62,399	InnoDB	latin1_swedish_ci	3.5 MiB	
Mtruncatula_Panther_Mt4_0v1	Browse Structure Search Insert Empty Drop	~62,156	InnoDB	latin1_swedish_ci	4.5 MiB	
Mtruncatula_PeptideSequence_Mt4_0v1	Browse Structure Search Insert Empty Drop	~62,031	InnoDB	latin1_swedish_ci	26.6 MiB	
Mtruncatula_Pfam	Browse Structure Search Insert Empty Drop	~62,298	InnoDB	latin1_swedish_ci	4.5 MiB	
Mtruncatula_ProteinAnnotation_Mt4_0v1	Browse Structure Search Insert Empty Drop	~63,569	InnoDB	latin1_swedish_ci	4.5 MiB	
Mtruncatula_SearchTableID_Mt4_0v1	Browse Structure Search Insert Empty Drop	~68,839	InnoDB	latin1_swedish_ci	3.5 MiB	
Table_name	Browse Structure Search Insert Empty Drop	~8	InnoDB	latin1_swedish_ci	16 KiB	

Figure 32. Mtruncatula DB tables

All these essential files are imported as tables to the KBC_Mtruncatula database. This organism will be displayed on the Browse KBCCommons section on the KBCCommons

home page. It will also be available for selection under Contribute to KB and Add version to KB sections as this organism's KB has been already established with the 6 essential files.

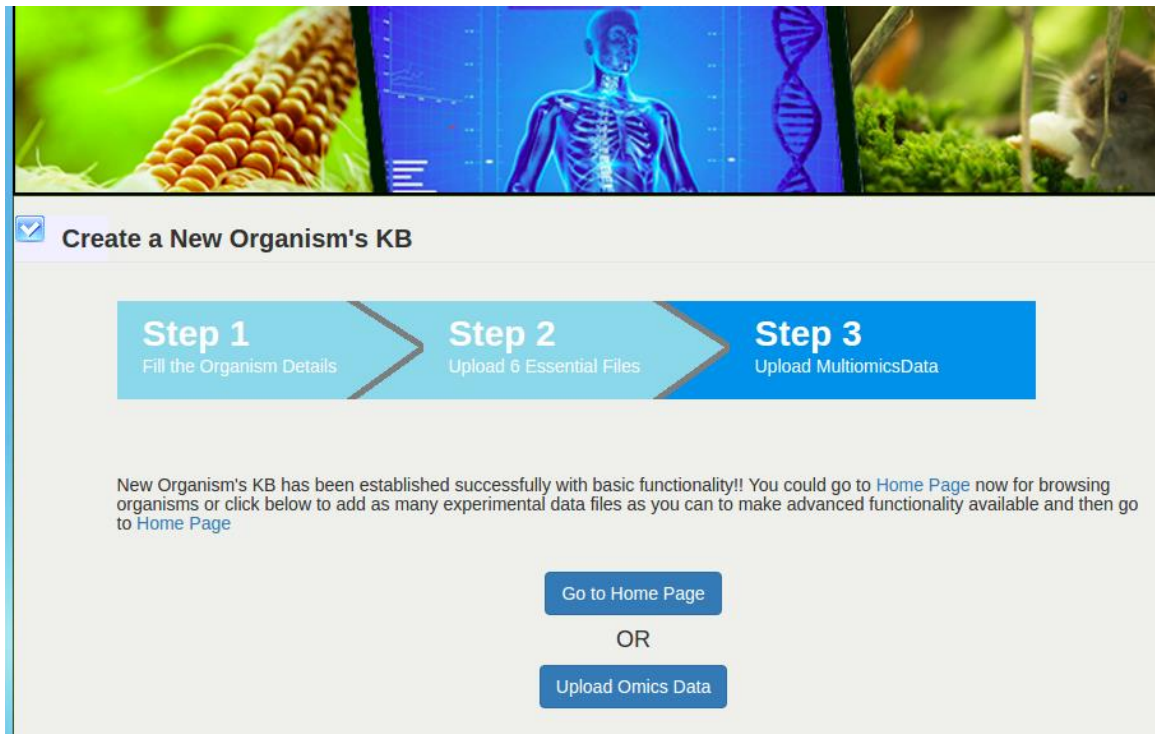


Figure 33. KBCOMMONS newkb steps-omics option

After the 6 essential files' are processed and imported into the database then an option to go to step 3 will be shown. Which will take users to this page shown in Figure 33, where they can either browse this organism by going to the KBCOMMONS home page or contributing further by uploading multi-omics or entities data types. If user wants to browse with these uploaded essential files then go to KBCOMMONS home page and browse which is explained in *section 4.4* under Browse KBCOMMONS, if user wants to upload then click on selecting details of the data button which will take user to Add multi-omics data which is same as Figure 37(omics data selection) and explained in *section 4.3* under Contribute to KB.

4.3 Contribute to *Medicago truncatula* KB

In the Create a new KB step if the user could not upload multi-omics files or entities data then the user could stop the Create a new KB process and upload at a later time when user have corresponding data using this Contribute to KB option. The first step here is to select organism details from the existing list of organisms for which user wants to upload multi-omics data for any of the supporting data types of for the entities (Table 1).

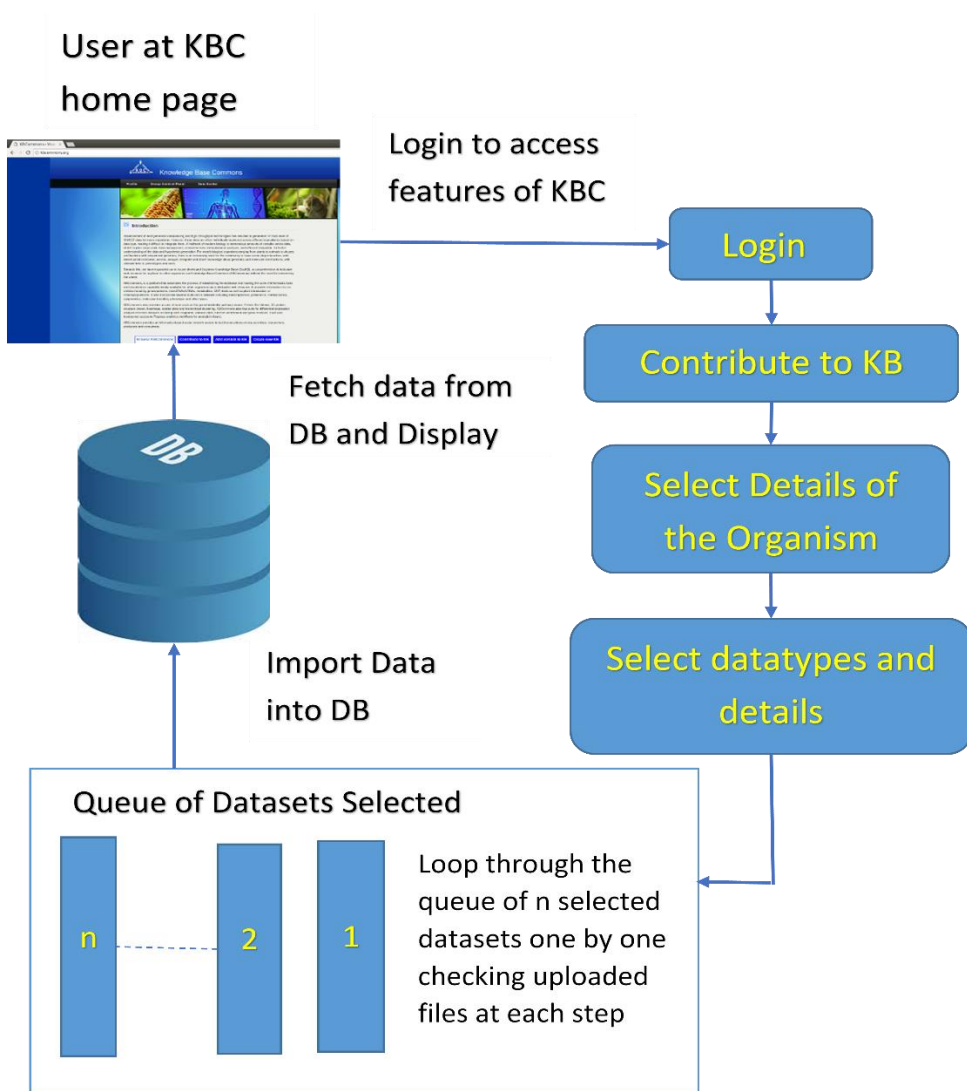


Figure 34. Contribute to KB workflow

A new KB for *Mtruncatula* has been established in the Create a new KB section with 6 essential files only, now we will continue contributing to this KB with omics data.

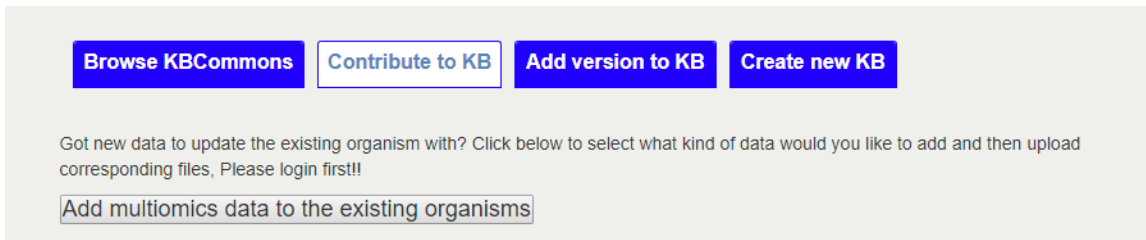


Figure 35. Contribute to KB button

This will be taken the user to the organism details selection page, which is the first step:

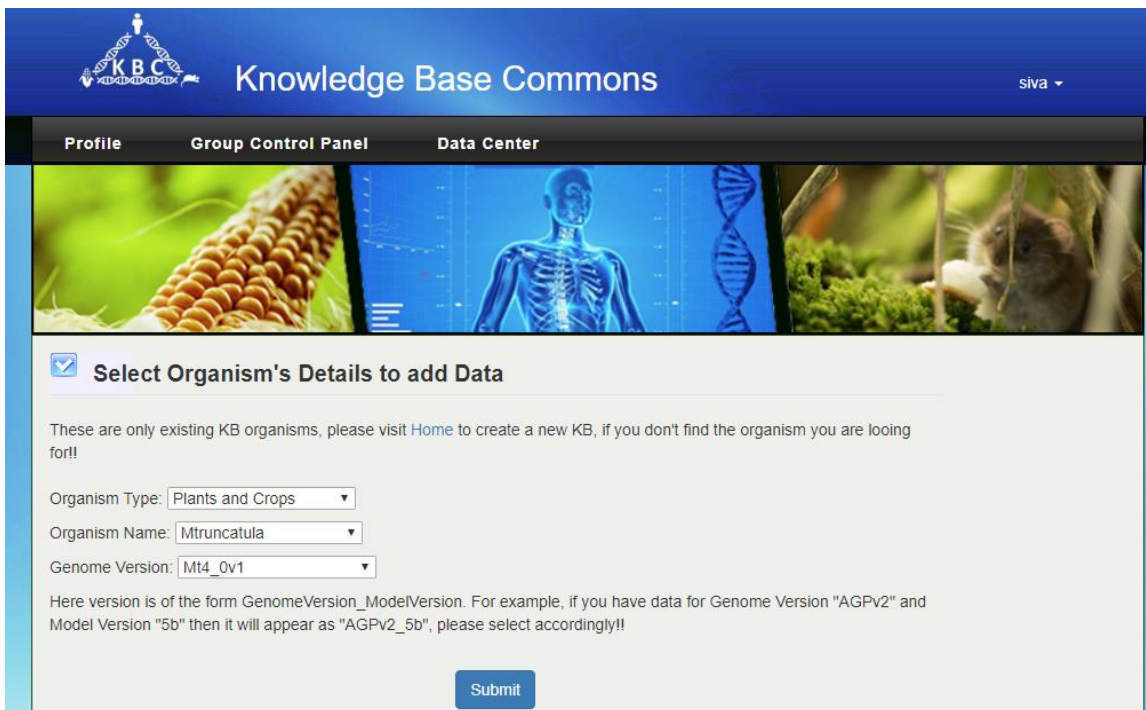


Figure 36. Select details for *Mtruncatula*

In this step to select details of *Mtruncatula*, we show organisms for which KB has been already established because only to that user could bring new data to update current KB. We give a link to homepage in case user could not find organism of interest.

Please select number of datasets you want to upload for each data type, leave it 0 if you do not have that particular data

Multiomics Data

Transcriptomics:

Details for RNA-Seq data:

Number of Gene Expression (GE) datasets:

Differential Expression Genes (DEG) data:

Details for Microarray data:

Number of Gene Expression (GE) datasets:

Differential Expression Genes (DEG) data:

Proteomics:

Details for Mass Spectrometry data:

Number of Protein Expression (PE) datasets:

Differential Expression Protein (DEP) data:

Details for 2DGel data:

Number of Protein Expression (PE) datasets:

Differential Expression Protein (DEP) data:

Epigenomics:

Details for Bisulphite Sequencing data:

Number of Methylation datasets:

Differential Methylated Regions (DMR) data:

Details for Methylation Array data:

Number of Methylation datasets:

Differential Methylated Regions (DMR) data:

Details for MBD-Seq Array data:

Number of Methylation datasets:

Differential Methylated Regions (DMR) data:

Entities

- miRNA/sRNA
- Metabolite
- SNP-Insertion/deletion
- Plant Introduction Lines/Animal Strains
- Phenotype/TRAIT/Disease

Submit

Figure 37. Omics and Entities data options

The user will enter as many data sets as they have for each data type and leave it zero if they do not have data for that particular data type. In this example, we have Transcriptomics RNA-Seq Gene expression with FPKM datatype and differential expression data with cuffdiff methods. This data has 8 conditions and 3 replicates. We also have a Proteomics data with the 2DGel method and Epigenomics data with Bisulphite Methylation method and an SNP entity dataset. All these demo files for illustrating with *Medicago truncatula*. Here we selected total 4 datatypes and one dataset for each datatype hence it is a loop of four web pages in automatic stepwise fashion starting with transcriptomic RNA-Seq and ends with SNP entity; after SNP dataset is uploaded loop will be terminated and will be directly taken the user to the KBCommons homepage.

You are selecting details for dataset 1 of total selected 4 datasets
Enter Details for Dataset 1 for Transcriptomics_RNA-Seq:

Dataset type: ReadCount FPKM/RPKM
Dataset privacy: Public Private
Number of Conditions:
Number of Replicates:
Name of the Dataset:
Method of the Data:
#Differential Expression Genes(DEG):
Number of DEG files:

Figure 38. RNA-Seq dataset details selection-1

On each page index of the dataset will be displayed to keep track of the loop. Here we have four datasets and this is our first dataset to start with. At first, user has to select details about RNA-Seq dataset and number of differential expression files they have and upon

confirming this conditions for file checks and options to upload files for each data type will be appearing on the same page.

Gene Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: number of conditions * number of replicates +1
- First column name should be GeneID
- All gene ids in the file must be from the selected organism and genome version
- Only integers for ReadCount method and Decimals for FPKM method
- No empty values are allowed in the file

Choose Gene Expression File:

Demo_Trans..._FPKM.csv

Name of the Differential Expression should follow this format, Condition1_vs_Condition2, because Fold Change(FC) is calculated as(Condition2/Condition1)

Differential Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: 12
- First column name should be GeneID
- All gene ids in the file must be from the selected organism and genome version

Details for DEG #1

Name:

Method:

File: Pana_Dr_Sus...Sus_dr.csv

Details for DEG #2

Name:

Method:

File: Pana_Dr_Sus...ol_ctrl.csv

Details for DEG #3

Name:

Method:

File: Pana_Dr_Sus...ol_ctrl.csv

Details for DEG #4

Name:

Method:

File: Pana_Dr_Sus...Tol_dr.csv

Figure 39. RNA-Seq dataset file uploads-2

In RNA-Seq expression data for this demo, we have four different comparisons between conditions and hence we display options for each differential expression files to select and details to give. After filling details for each dataset and selecting the datasets; our index in the loop will be incremented by one and it will go to the next selected datatype after importing files to the database at the backend upon clicking submit.

You are selecting details for dataset 2 of total selected 4 datasets
Enter Details for Dataset 1 for Proteomics_2DGel:

Dataset privacy: Public Private

Number of Conditions:

Number of Replicates:

Name of the Dataset:

Protein Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: number of conditions * number of replicates +1 and can have four extra columns at the end including(same order): Peptide, Molecular Weight, Retention Time, Spot ID
- First column name should be "GeneID"
- All gene ids in the file must be from the selected organism and genome version

Choose Protein Expression File:

Demo_Prote..._2DGel.csv

Figure 40. 2DGel dataset selection

As we can see on this page it is displayed dataset 2 of total selected 4: in our dataset selection page we selected total 4 and the 2DGel dataset is our second dataset. It has five seed conditions with no replicates in the data. In this example, we do not have differential proteomic data and hence we will not show options for differential proteomics.

Our next datatype is Bisulphite sequencing methylation data with four conditions and no replicates. This is our third dataset in the loop and will display as dataset 3 of total selected 4. Methylation dataset type can be CG, CHG, and/or CHH because *Mtruncatula* belongs to Plant organism, otherwise, we will give only CG option.

You are selecting details for dataset 3 of total selected 4 datasets

Enter Details for Dataset 1 for Epigenomics_BisulphiteSequencing:

Dataset type: CG CHG CHH Select data types that you have data for

Dataset privacy: Public Private

Number of Conditions:

Number of Replicates:

Name of the Dataset:

Methylation File must follow some conditions for successful File Checks, We allow two different file formats:

- **If file first column is chromosome_name then follow below checks:**
- File should have these columns including: chromosome_name, start_position, end_position, datatype, and conditions * replicates
- Total number of columns must be equals to: number of conditions * number of replicates + 4 as mentioned above
- First four columns names should be in the order
- **If file first column is GeneID then follow below checks:**
- File should have these columns including: GeneID, number of datatypes (checked above) * number of conditions * number of replicates: for example if user checked only for CG and CHG then number of datatypes would be 2
- First column should be GeneID
- All gene ids in the file must be from the selected organism and genome version

Choose Methylation File:

Demo_methyl...ulphite.csv

Figure 41. Bisulphite Sequencing data selection

We allow two different formats of Bisulphite sequencing data which can have chromosome information or GeneID information. With GeneID information in our data we will have a number of datatypes, here we have all three (CG, CHH, CHG), (3) * number of conditions (4) * number of replicates (1) + GeneID (1) = 13 columns.

Our final dataset in this example is SNP entity which can again have three SNP datatypes including SNP Array, GWAS, and SNP Dinucleotide or Insertion and Deletion options. The user can select as many datasets as they have for each datatype and confirm the number to display options to select files.

You are selecting details for dataset 4 of total selected 4 datasets
Enter Details for SNP for each datatype :

SNParray
Number of SNParray datasets:

GWAS
Number of GWAS datasets:

SNP dinucleotide
Number of SNP dinucleotide datasets:

Insertion/Deletion polymorphism (indel)
Number of indel datasets:

Please select files for selected datasets below:

File for GWAS #1 dataset
File:
 Demo_gwas.csv

File for indel polymorphism #1 dataset
Mention insertion or deletion in the file
File:
 Demo_Indel.csv

Figure 42. SNP data selection

This is the final dataset hence it is displayed as dataset 4 of totally selected 4. Hence after uploading files and proper file checks user will be automatically redirected to the KBCommons home page upon clicking submit button. All these datasets are converted to

tables at the backend and details of tables are stored in the reference table. Below are the two backend table snippets of gene expression and differential expression for Transcriptomics data that we uploaded on the browser.

GeneID	Pana_Dr_Sus_ctrl_R1	Pana_Dr_Sus_ctrl_R2	Pana_Dr_Sus_ctrl_R3	Pana_Dr_Sus_dr_R1	Pana_Dr_Sus_dr_R2	Pana_Dr_Sus_dr_R3	PI567690_Dr_Tol_ctrl_R1	PI567690_Dr_Tol_ctrl_R2
Medtr1g004930.1	1.17381000	2.21863000	3.95371000	4.41499000	4.31659000	3.68594000	3.26938000	2.99961000
Medtr1g004940.1	10.28970000	12.94200000	12.74410000	4.05048000	9.85082000	10.85530000	7.84441000	9.82046000
Medtr1g004950.1	0.19741400	0.94672900	2.26714000	3.99269000	3.65098000	2.40080000	2.16058000	2.73138000
Medtr1g004960.1	1.92651000	7.49857000	7.68913000	12.36030000	8.99994000	8.74919000	12.73200000	13.84930000
Medtr1g004980.1	0.00000000	0.07731050	0.30064100	0.57822100	0.39080800	0.25308100	0.00000000	0.10162500
Medtr1g004990.2	0.00000000	0.05056500	0.03932690	0.07563720	0.08520280	0.13242200	0.06581640	0.13293600
Medtr1g004990.1	0.00000000	0.06742120	0.13983200	0.47064000	0.22721200	0.19128000	0.02925230	0.10339700
Medtr1g006460.1	0.00000000	0.04938040	0.26883900	0.00000000	0.24962000	0.19398000	0.06427450	0.00000000
Medtr1g006490.1	1.50366000	3.77237000	3.83672000	4.99176000	5.01186000	3.79973000	3.58821000	7.15213000
Medtr1g006530.1	1.67593000	1.70820000	4.52783000	7.08317000	5.92072000	4.16927000	3.76022000	4.02860000
Medtr1g006540.1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Medtr1g006590.1	0.00000000	0.56384000	3.28895000	1.26512000	0.95008000	1.84577000	2.56867000	2.59411000

Figure 43. GeneExpression RNA-Seq table

GeneID	fold_change	locus	p_value	q_value	sample_1	sample_2	significant	status	test_stat	value_1	value_2
Medtr1g004930.1	0.42457900	Gm02:2248-17672	0.35125000	0.65889000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	0.55188500	2.76833000	3.71560000
Medtr1g004940.1	-0.91070900	Gm03:8648-28244	0.05440000	0.26963000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	-1.27497000	14.31840000	7.61628000
Medtr1g004950.1	1.26854000	Gm05:16184-21667	0.02650000	0.18618000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	1.31913000	1.23252000	2.96935000
Medtr1g004960.1	0.46956700	Gm06:1757-15096	0.31655000	0.62945300	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	0.63715200	6.50280000	9.00439000
Medtr1g004980.1	1.44773000	Gm07:1209-3554	1.00000000	1.00000000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	NOTEST	0.00000000	0.13166000	0.35913800
Medtr1g004990.2	1.43242000	Gm10:10493-13224	1.00000000	1.00000000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	NOTEST	0.00000000	0.03356580	0.09059400
Medtr1g004990.1	1.82485000	Gm11:1119-5508	1.00000000	1.00000000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	NOTEST	0.00000000	0.07394810	0.26197500
Medtr1g006460.1	0.31059700	Gm15:626-2726	1.00000000	1.00000000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	NOTEST	0.00000000	0.10975300	0.13611700
Medtr1g006490.1	0.22729400	Gm17:405-1845	0.74530000	0.89866400	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	0.33640700	3.51191000	4.11117000
Medtr1g006530.1	0.76543300	Gm18:2484-40137	0.09705000	0.35948200	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	0.98132100	2.98972000	5.08216000
Medtr1g006540.1	0.00000000	scaffold_21:663659-6	1.00000000	1.00000000	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	NOTEST	0.00000000	0.00000000	0.00000000
Medtr1g006590.1	-0.07841850	scaffold_21:684712-6	0.94395000	0.98013700	Pana_Dr_Sus_ctrl	Pana_Dr_Sus_dr	no	OK	-0.10765000	1.32643000	1.25625000

Figure 44. Differential Expression Cuffdiff table

In the above table of differential expression, we have sample_1 and sample_2 columns which represent condition1 and condition2 for which we have this differential expression data.

In this example, I introduced only a few datatypes functionality and how we could proceed with how many ever datasets we have and there are no restrictions on the number of datasets to bring to contribute to existing KB. I will briefly show other datatypes and their file checks for transcriptomic microarray data, proteomic mass spectrometry data, epigenomic methylation array and MBD-Seq array data. Methylation array and MBD-Seq

options and file condition look similar hence I will show for methylation array. I will also show for other entities including miRNA/sRNA and Metabolite.

You are selecting details for dataset 1 of total selected 1 datasets

Enter Details for Dataset 1 for Transcriptomics_Microarray:

Dataset type: Affymetrix Agilent
Dataset privacy: Public Private
Number of Conditions:
Number of Replicates:
Name of the Dataset:
Method of the Data:

#Differential Expression Genes(DEG):

Number of DEG files:

Gene Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: number of conditions * number of replicates +2
- First and second columns name should be GeneID and Affy_Probe_ID
- All gene ids in the file must be from the selected organism and genome version
- No empty values are allowed in the file

Choose Gene Expression File:

Demo_microarray.csv

Name of the Differential Expression should follow this format, Condition2_vs_Condition1, because Fold Change(FC) is calculated as(Condition2/Condition1)

Differential Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: 12
- First column name should be GeneID
- All gene ids in the file must be from the selected organism and genome version

Details for DEG #1

Name:
File: Demo_microarray_DE.csv

Figure 45. Transcriptomic Microarray selection details

In transcriptomic microarray selection, we currently support two different dataset types Affymetrix and Agilent. For each type, we have two methods including RMA and Lowees. In addition to the GeneID column which is necessary for most of the files, for this file type, we also search for Affy_Probe_ID column. Based on the dataset type we check

the selected file for correctness, if files satisfy all conditions then we will import these files to the database at the backend upon clicking submit button.

You are selecting details for dataset 1 of total selected 1 datasets
Enter Details for Dataset 1 for Proteomics_MassSpectrometry:

Dataset type: GCMS LCMS
Dataset privacy: Public Private
Number of Conditions:
Number of Replicates:
Name of the Dataset:

#Differential Expression Protein (DEP):
Number of DEP files:

Protein Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: number of conditions * number of replicates +1 and can have four extra columns at the end including(same oder): Peptide, Molecular Weight, Retention Time, Spot ID
- First column name should be GeneID
- All gene ids in the file must be from the selected organism and genome version

Choose Protein Expression File:
 Demo_massSpec.csv

Name of the Differential Expression should follow this format, Condition2_vs_Condition1, because Fold Change(FC) is calculated as(Condition2/Condition1)

Differential Expression File must follow some conditions for successful File Checks including:

- All gene ids in the file must be from the selected organism and genome version

Details for DEP #1
Name:
File: Demo_massSpec_DE.csv

Figure 46. Proteomic MassSpectrometry selection details

In Proteomic MassSpectrometry selection we currently support two different dataset types GCMS and LCMS. Before submit button appears user has to enter a number of conditions and replicates and a number of differential expression files and confirm them. If the number of differential expression compatible with number of conditions and replicates then options to select files and to enter details will appear along with submit

button, if conditions fail then a descriptive error message with information about which condition failed will be displayed on the same page to allow user to update the file accordingly and upload again.

You are selecting details for dataset 1 of total selected 1 datasets

Enter Details for Dataset 1 for Epigenomics_MethylationArray:

Dataset type: CG CHG CHH Select data types that you have data for

Dataset privacy: Public Private

Number of Conditions:

Number of Replicates:

Name of the Dataset:

#Differential Methylated Regions(DMR):

Number of DMR files:

Methylation File must follow some conditions for successful File Checks including:

- File should have these columns including: GeneID, probe_id, datatype, and number of conditions * number of replicates
- First three columns should follow the above mentioned order
- All gene ids in the file must be from the selected organism and genome version

Choose Methylation File:

Demo_methyl...nArray.csv

Name of the Differential methylated region should follow this format, Condition2_vs_Condition1, because Fold Change(FC) is calculated as(Condition2/Condition1)

Differential methylated File must follow some conditions for successful File Checks including:

- Should have 8 columns in this order: chromosome_name, start_position, end_position, logFC, logCPM, LR, PValue, FDR

Details for DMR #1

Name:

File:

Demo_methy...ray_DE.csv

Figure 47. Methylation Array selection details

Options and file checks for Methylation array dataset and MBD-Seq dataset are almost the same hence I showed one of them. For methylation data, if an organism belongs to Plants and Crops then we will have all three datatypes CG, CHH, CHG if non-plant organism then only CG option will be displayed.

You are selecting details for dataset 1 of total selected 1 datasets
Enter Details for miRNA for each datatype :

miRNA Expression

Number of miRNA expression datasets:

miRNA - Target gene

Number of Target gene datasets:

Differential Expression miRNA

Number of Differential Expression datasets:

Please select files for selected datasets below:

File for miRNA Expression #1 dataset

File:

No file chosen

File for Target gene #1 dataset

File:

No file chosen

File for Target gene #2 dataset

File:

No file chosen

File for Differential Expression #1 dataset

File:

No file chosen

Figure 48. miRNA files details

For miRNA datatype we are supporting three different datatypes including miRNA expression data, miRNA – Target genes data, and Differential expression miRNA data.

Enter Details for Dataset 1 for Metabolite:

Dataset type: GCMS LCMS
 Dataset privacy: Public Private
 Number of Conditions:
 Number of Replicates:
 Name of the Dataset:

#Differential Expression Metabolite (DEM):

Number of DEM files:

#Metabolite Annotation File: Yes No

Metabolite Expression File must follow some conditions for successful File Checks including:

- Number of columns must be equals to: number of conditions * number of replicates +1 and can have two extra columns including(same order): Mass to Charge ratio and Retention Time
- First column name should be Metabolite Name

Choose Metabolite Expression File:

No file chosen

Name of the Differential Expression should follow this format, Condition2_vs_Condition1, because Fold Change(FC) is calculated as(Condition2/Condition1)

Details for DEM #1

Name:

File: No file chosen

Choose Metabolite Annotation File

File: No file chosen

Annotation file can have columns including Metabolite Name, ID, Formula(SMILES), Structure, and KEGG Compound ID

Figure 49. Metabolite datatype options

In metabolite entity datatype we are supporting three types of files including Metabolite expression file, Differential Expression Metabolite File, and an optional Metabolite Annotation File. With the Metabolite Annotation file, user can provide us with details about the metabolites including their SMILES chemical formula, structure, and a universal KEGG compound ID which can be used to map to the other databases.

These are the supported datatypes and possible file checks that we do in order to import data to the backend tables and to perform data analysis with correct files.

4.4 Browse KBCommons for *Medicago truncatula*

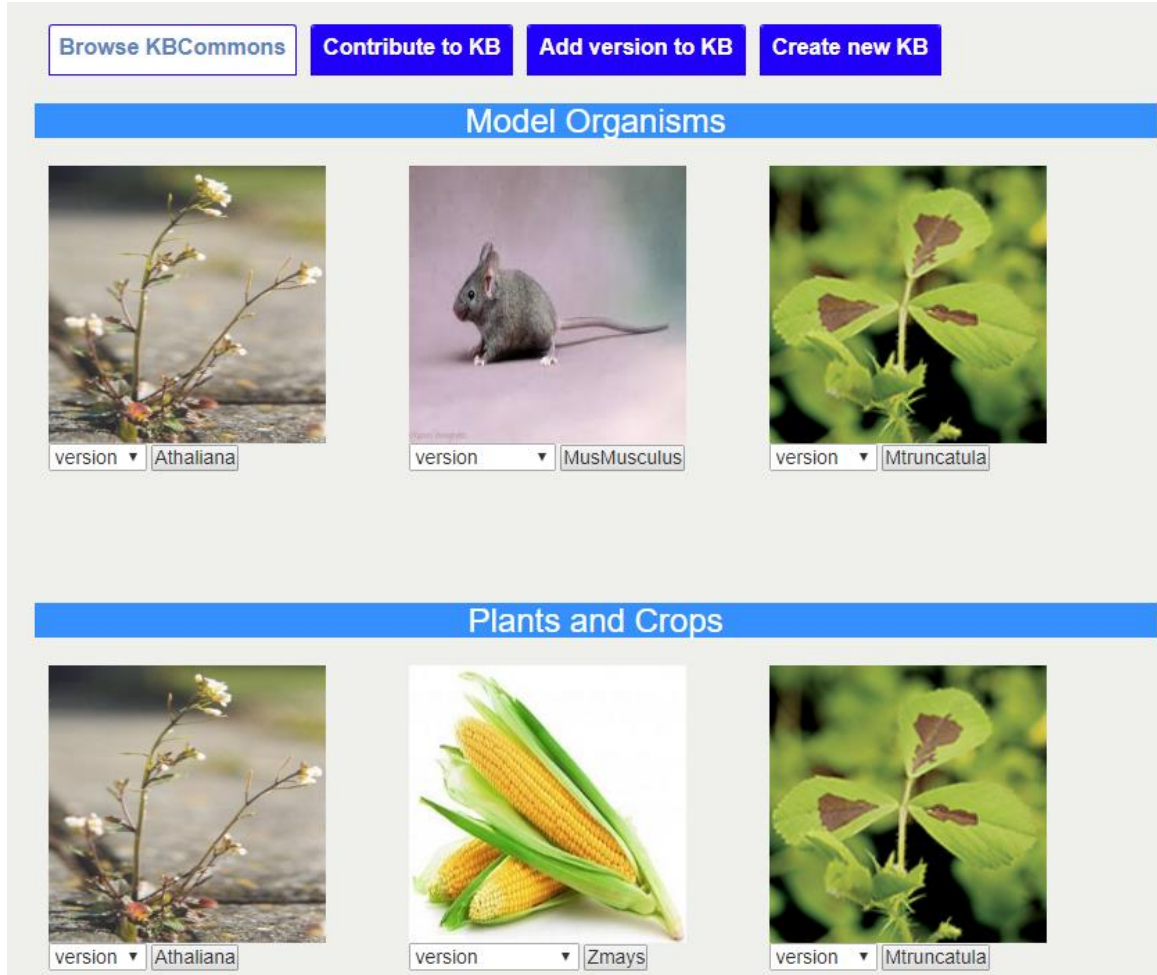


Figure 50. *Mtruncatula* in Browse KB

Mtruncatula has been added to both Model Organisms category and Plants and Crops category. Now user can click on *Mtruncatula* to browse the uploaded data on its homepage, Figure 51. User can search gene ids of *Mtruncatula*, and can see all the details of that gene id. Example gene ids based on the selected version will automatically be displayed on the Gene Search page, shown in Figure 52. We show chromosomal and sequence information of the selected gene id on gene card page, shown in Figures 53, 54, and 55.

We also display differential expression data in browse section. We show existing versions of the selected organism and existing differential expression data along with sample name provided and the method of the data. We display all combinations of conditions for which user uploaded differential data with an option to select any of their combinations. We have options to select whether the user wants to see data only for Up-regulated or Down-regulated or for both. After data options selection and submitting we show results in plain text format and also in the pie chart for better visualization to understand data better, shown in Figures 56 and 57.

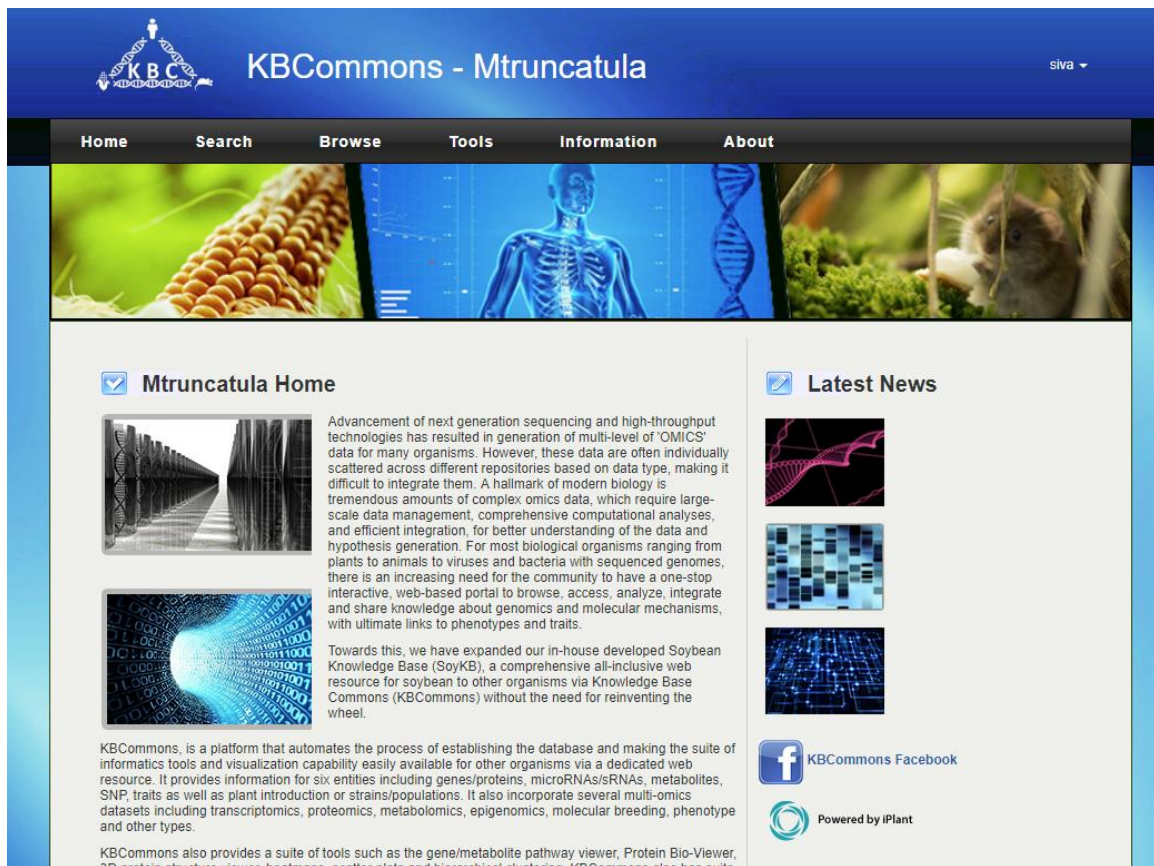


Figure 51. Mtruncatula home page

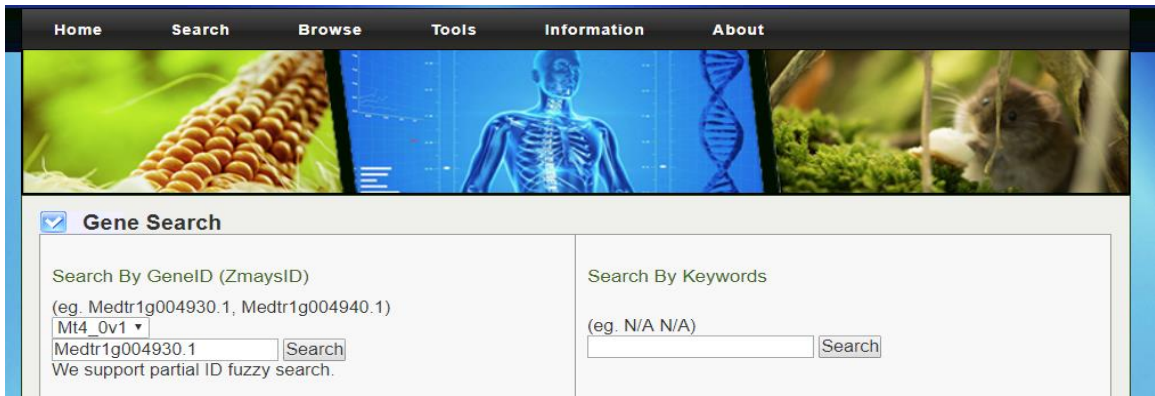


Figure 52. *Mtruncatula* Search Gene ID

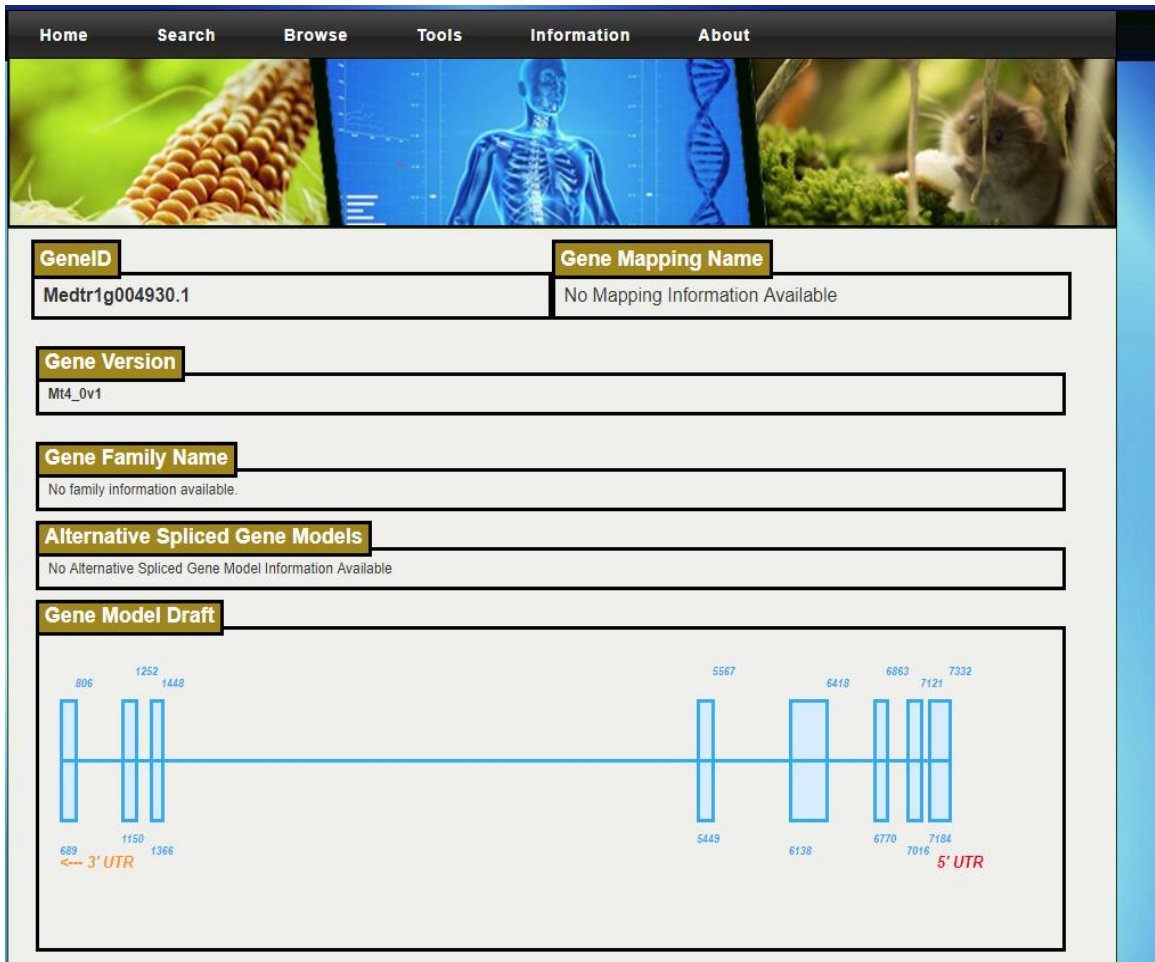


Figure 53. *Mtruncatula* gene card page

Chromosomal Information							
Gene	Chromosome	Start	End	Strand	PACID	LetterCode*Note	GenomeBrowser
mRNA	chr1	689	7332	-	31100295	N/A	protein_coding_gene See!
Exons							
CDS	chr1	7184	7332	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	7016	7121	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	6770	6863	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	6138	6418	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	5449	5567	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	1366	1448	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	1150	1252	-	31100295	N/A	protein_coding_gene See!
CDS	chr1	689	806	-	31100295	N/A	protein_coding_gene See!

Figure 54. *Mtruncatula* Search Gene ID Chromosomal Information

Sequence Information	
cDNA 1053 bp	<pre> ATGATTATATCATTACTCCCTTGTAATCAGCTGATAAACCATCAGGCCGTAGGTCAGCC CAAGCTTTTATTTGGAAGCCTTTAATGGTAAGATTGCAAGCTTCCGGTGCTGTTTCTTCA TATTCCTCAGGAACATTGATAGGACATAGCCTCCGAAAGTTTCGACTCTATCGCCTACTCA GAACATGAGTCTTTGGTGAAGTGTAGCAGCTACACATTCACTGGTATGTATCCACTCTTT GTTGAGGATGATAGAAGTTTTTCAGTTTTAATTGAACGAATTGCAAAGCTTGGAAATCATC AACAAACATTCTCGGGGGAAGGAGGACGTTGAGAAAGAAGCTGTAAAAGTCACTGTCAAAA CGATTGGGCACTCGATTCAAAATTGCACCCCTATTTTTAAAAATGTGCGCAAGGCCGCAAG CAAAGGATGATAAGGGGAAACAACCAGTAGTGAACAAGGCTGGTCCATTGATGATGAGA GTATTCTTACAACCTGCCGGCATCAATGACGACTCCCACCTTTACTGATGGTAGTGAAGAG TGTTTTATTCCAGGCAATCAAGTTTCGGAACACCAGTTGAGGTGCGACCAGTCCAGTT TTAAGCAAAGTGGTGCTAGACGACATTGCGTTTTCAATCTTTCAACGGTCAGAAAAGAA GGAACGCAGAGAAAATACCTGTCTTAATTTAGATTTTGTAGAGTTACTTGACGCCCA AGGATAGTGATAAAATAAACTTCTTGTAGTTTGTGCACCGTGATTTGAATGTGGATGAT TTGAACAGTTTGATACAAAGTTTTCTTAAGATATTATTATCGTATCTTATTGTCATCAAG CTTATGGCATCCATTATCAAAATTGATTCAATCTGGATTGATTTTTCCCAATACGATTTT AAGGATCCAGACCAATGTGAACCTTCAATTATCAGGTCCTTGGTTACTGGTGGGACCAAG GTTGCAAGTATTTGTCGAGTGTCTCCACTTAGTTTGAAGAGTTGGTCTCCCTCAAAGT GCAAGTTTTCTCCTCGAACTTCTCCTGAATAA </pre>
cds 1053 bp	<pre> ATGATTATATCATTACTCCCTTGTAATCAGCTGATAAACCATCAGGCCGTAGGTCAGCC CAAGCTTTTATTTGGAAGCCTTTAATGGTAAGATTGCAAGCTTCCGGTGCTGTTTCTTCA TATTCCTCAGGAACATTGATAGGACATAGCCTCCGAAAGTTTCGACTCTATCGCCTACTCA GAACATGAGTCTTTGGTGAAGTGTAGCAGCTACACATTCACTGGTATGTATCCACTCTTT GTTGAGGATGATAGAAGTTTTTCAGTTTTAATTGAACGAATTGCAAAGCTTGGAAATCATC AACAAACATTCTCGGGGGAAGGAGGACGTTGAGAAAGAAGCTGTAAAAGTCACTGTCAAAA CGATTGGGCACTCGATTCAAAATTGCACCCCTATTTTTAAAAATGTGCGCAAGGCCGCAAG CAAAGGATGATAAGGGGAAACAACCAGTAGTGAACAAGGCTGGTCCATTGATGATGAGA GTATTCTTACAACCTGCCGGCATCAATGACGACTCCCACCTTTACTGATGGTAGTGAAGAG TGTTTTATTCCAGGCAATCAAGTTTCGGAACACCAGTTGAGGTGCGACCAGTCCAGTT TTAAGCAAAGTGGTGCTAGACGACATTGCGTTTTCAATCTTTCAACGGTCAGAAAAGAA GGAACGCAGAGAAAATACCTGTCTTAATTTAGATTTTGTAGAGTTACTTGACGCCCA AGGATAGTGATAAAATAAACTTCTTGTAGTTTGTGCACCGTGATTTGAATGTGGATGAT TTGAACAGTTTGATACAAAGTTTTCTTAAGATATTATTATCGTATCTTATTGTCATCAAG CTTATGGCATCCATTATCAAAATTGATTCAATCTGGATTGATTTTTCCCAATACGATTTT AAGGATCCAGACCAATGTGAACCTTCAATTATCAGGTCCTTGGTTACTGGTGGGACCAAG GTTGCAAGTATTTGTCGAGTGTCTCCACTTAGTTTGAAGAGTTGGTCTCCCTCAAAGT GCAAGTTTTCTCCTCGAACTTCTCCTGAATAA </pre>
Peptides	<pre> MIISLLPKSADKPSGRSAQAFIWKPLMVRLQASGAVSSYSSGTLIGHSLRSFDSIAYS EHESLVKSSYFTFGMYPLFVEDDRSFSVLIERIAKLGIIINHRSRGKEDVEKEAVKVTVK RLGTRFKIAPLFLKCAQGRKQKDDKQKQPVVNKAGPLMMRVFLQLPASMTTPTFTDGSSE </pre>

Figure 55. *Mtruncatula* Search Gene ID Sequence Information

Differential Expression

Please select the gene version, data type, dataset and method

Mt4_v1 Differential transcriptomics Demo_Mtruncatula cuffdiff

Pana_Dr_Sus_ctrl_vs_Pana_Dr_Sus_dr Pana_Dr_Sus_ctrl_vs_PI567690_Dr_Tol_ctrl
 Pana_Dr_Sus_dr_vs_PI567690_Dr_Tol_ctrl Pana_Dr_Sus_dr_vs_PI567690_Dr_Tol_dr

Please set up filtering parameters:

Fold change method: log2(x): 2

Absolute log value: 1

Filter value:

q-value: 0.05

Analysis options:

Figure 56. Differential expression data retrieval

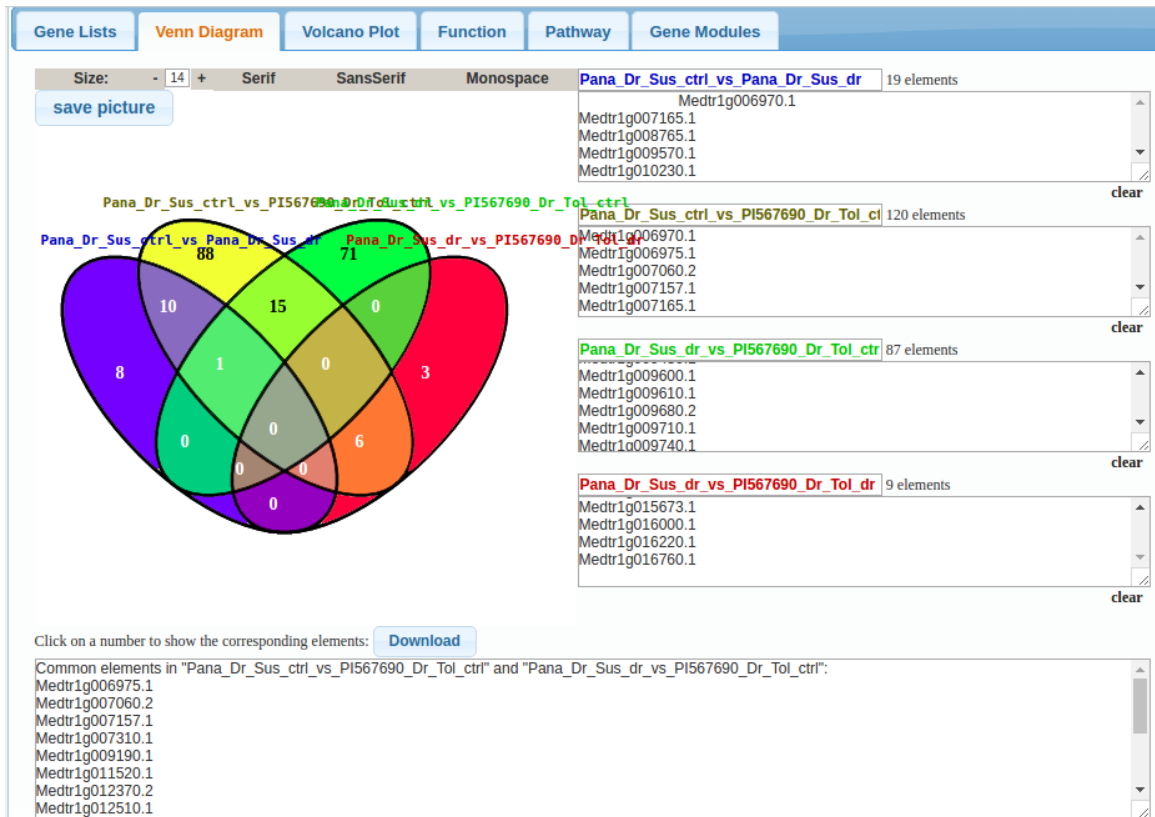


Figure 57. Differential expression data displayed in Venn diagram

4.5 Add version to KB for *Medicago truncatula*

If the user wants to add another version to the existing organism's KB then there is an option to do this. The procedure to this is similar to creating a new KB with fewer options to select details of the organism. In this section, we display only existing organism to select to add a new version.

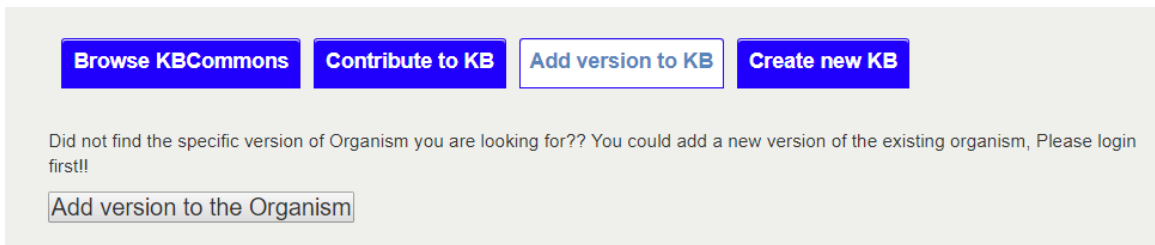


Figure 58. Add version to KB button

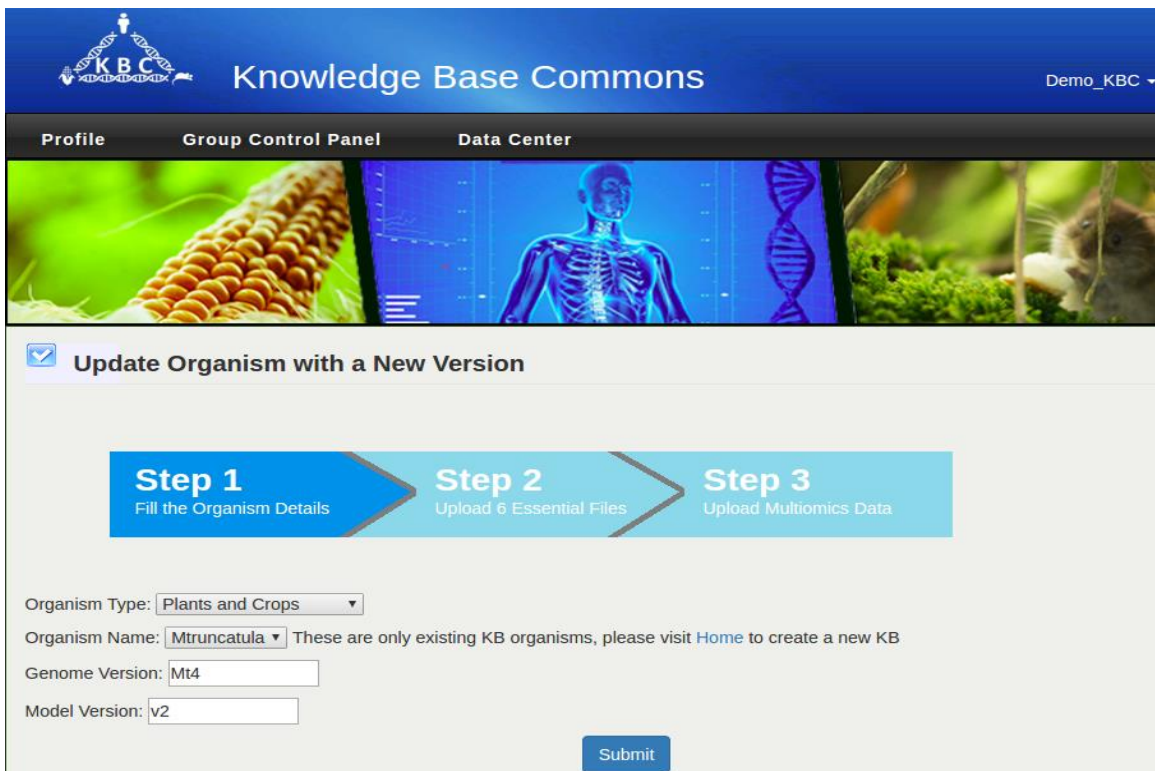


Figure 59. Add version to KB details

5. Testing

Testing is one of the important stages in the Software Development Life Cycle. It helps to identify defects and errors in the model and guide us to improve the quality of the development. It also helps to make sure the reliability and satisfaction of the end user. There are many types of testing including: Unit testing, Integration testing, Module testing, Sub-System testing, System testing, Acceptance testing, Performance testing, Functional testing, and so on.

A brief introduction to some of the testing that has been done on this KBCCommons development:

- **Unit Testing:** It is the testing of an individual unit. Each unit is tested independently, without other system components. This helps to identify syntax and logic errors from a single unit. Individual components are tested to ensure they operate correctly.
- **Functional Testing:** It is the testing to ensure that specified functionality works well based on the system requirements.
- **Sub-System Testing:** This tests collection of integrated modules, which have been developed separately and combined later into a sub-system. We test for component interactions in this type of testing.
- **System Testing:** This is testing the entire system after integrating all the developed sub-systems into one main system. This helps to identify the unanticipated interactions among the integrated sub-systems and to test the functional and non-functional requirements. We also test for compatibility of the software in different environments.

➤ **Performance Testing:** It is the testing to access the speed and effectiveness of the system and to check if the system is generating results within the specified time.

Our development has been tested with most of the above mentioned types of testing. Unit testing has been done on Create a New KBCommons feature, Upload MultiOmics files feature, and Add version to KB feature independently and they all are worked as shown in the Case Study for *Mtruncatula*.

We also tested combining the Create a New KBCommons feature and Upload MultiOmics files feature to test the compatibility of combining the different modules. They worked well without any issues. This ensures integrity of the components.

The entire system along with other components from other developers has been tested. For example, retrieving data after creating a new KB and using various analysis tools on uploaded data. This makes sure that the entire system is performing well.

Functional and Performance testing has been done on the Development from different users. Due to lack of omics data for organisms, I assigned testing creating a new KB feature to 4 of our lab students and based on the reports collected from them: it took around 3-4 minutes to upload and import files to the database on an average for a file size around 0.5GB.

From these results, we can see that this is a well working system with four features of Creating a new KB, Add version to KB, Uploading multiomics files, and Browsing existing data. However, there are many places where performance and quality can be

improved. Our system is flexible that, new improvements can easily be incorporated in to the existing development.

A more extensive testing is required to further ensure better quality of the development. For example, when user creates a new organism's KB then we are directly importing files to the database and making them available on the website. A brief review of the uploaded data is required to make sure that user uploaded good quality data.

6. Applications of KBCommons

KBCommons currently supporting organisms belong to four main categories including Plants and Crops, Animals and Pets, Humans and Diseases, Microbes and Viruses.

Studying organisms belong to Plants and Crops will be very useful not only to increase the yield of crops but also in precision agriculture and precision medicine. As we know that many plants and leaves can be used as medicine to some diseases as a natural remedy. A well-known and widely used medicine aspirin was developed as a result of the study of rotting tree bark. It is very important to study plants when using for medicine as plants have an impressive variety of defense mechanisms. Many leaves are poisonous to eat, some are poisonous to touch and a few species even provide nectar to insects that in turn defend the plant against predators.

Animal scientists can protect human health and it is important for scientists to study how diseases spread between humans and animals. In many parts of the world, animals are used as labor for farmers. Studying animals can be helpful to improve animal breeding, health, and nutrition. This can also be useful to keep our pets healthy. Studying this area can tackle issues with pets like pet obesity and breeding.

Precision medicine is an emerging approach for disease treatment and prevention that takes into account individual variability in genes, environment, and lifestyle for each person. Studying various diseases can be helpful in precision medicine and finally to improve human health and lifestyle. Understanding microbes can be useful in precision medicine too as they have a profound impact on every facet of human life and everything around us.

7. Summary

In this thesis, I have included five chapters of contents including Introduction to the biological and technical background, Database and frontend architecture, KBCommons functionalities, Case Study with an example, and Applications.

In the first chapter, I gave brief details about SoyKB and gaps of the SoyKB framework and why we started with developing a universal KBCommons framework for multiple organisms and what kind of functionalities we addressed in this first version. I also introduced technical perspective of this thesis and what are the technologies that I use for developing KBCommons.

In the second chapter, I explained the Frontend and Backend architecture and connectivity and how KBCommons implements automatics table generation and updating them with the data uploaded. Basically, I showed when user perform actions at the Frontend how we perform actions at the Backend using MySQL schemas presented using entity relationship diagrams.

In the third chapter, I talked about KBCommons workflow with a flowchart and what kind of data we are supporting in this version and what are the main sources of that data. I also talked about main functionalities of KBCommons, their workflow and required details that user has to bring.

In the fourth chapter, I explained KBCommons workflow with an example organism, *Medicago truncatula*, from user sign up to create a new KB and to browse uploaded data. In the final chapter of contents, I gave applications of KBCommons for each organism type that we addressed in this first version of KBCommons.

8. Future Study

KBCommons is an expansion to SoyKB, with a vision to make all possible features and tools available for users to use for many different organisms with the ultimate goal to do a genotype to phenotype conversion. In this thesis, we introduced our first version of KBCommons which addressed many of the features and tools that SoyKB has for soybean organism alone. In this first version of KBCommons, user can browse existing organisms' genes and differential expression and visualize in pie charts; user can create a totally new organism's KB with basic files and can also add multiple omics datasets and user can add a new version to the organism too with an automatic framework of creating tables at the backend and frontend web pages.

KBCommons development team is working towards addressing all the remaining features and tools that SoyKB has and our first version of KBCommons could not have including Pathway viewer, Affymetrix Probe ID mapper, Protein Bioviewer, Enrichment Analysis and options to download the existing data. In the multi-omics file uploads part, we do not have differential expression data for Proteomics and Epigenomics yet, our team continues to work to include these. We are also working on including other entities and enable file checks for them.

Applying data analytics across multiple organisms, also known as comparative genomics that is comparing one organism with another, is very important and useful to draw conclusions in various perspectives. Other members of KBCommons development team is working on implementing an application programmer interface (API) to enable this. It would be helpful if we have data analytics visualizations in various forms including 3D as the data scatter among different organisms.

SoyKB	Options	KBCommons_Status
Datatypes	Transcriptomics	Completed
	Proteomics	Completed
	Epigenomics	Completed
	miRNA/sRNA	Completed
	Metabolite	Completed
	SNP - Insertion/ Deletion	Completed
	Plant Introduction Lines / Animal Strains	In a few months
	Phenotype/TRAIT/Disease	In a few months
Search	Genes	Completed
	miRNA/sRNA	In a few weeks
	Metabolites	In a few weeks
	SNP	In a few weeks
	PI	In a few months
	TRAIT	In a few months
Browse	Gene Families	In a few weeks
	3D Protein Structure	In a few months
	Homeologous Genes	Soybean Specific
	Phosphorylation	In a few months
	G.Soja	Soybean Specific
	Genome Browser	In a few months
	<i>In Silico</i> Breeding Program	In a few months
	Fast Neutron Mutant Data	Soybean Specific
	Differential Expression	Completed
	eFP Browser	In a few months
	TF-ORFeome	Soybean Specific
	NGS Resequencing Browser	In a few months
Tools	Affymetrix Probe ID Mapper	In a few months
	Gene PathwayViewer	Completed
	MotifSampler Search with WebLogo	In a few weeks
	BLAST	Completed
	MULTIPLE SEQUENCE ALIGNMENT	In a few weeks
	Phylogeny	In a few weeks
	Chromosome Visualizer	In a few weeks
	Protein BioViewer	In a few months
	Heatmap	In a few weeks
	ScatterPlot	In a few weeks
	Targeted Sequence Selection	Soybean Specific
	SNPViz	In a few weeks
	Mutant Finder	Soybean Specific
	Enrichment Analysis	In a few months

Color	Description
Green	Completed
Yellow	In a few weeks
Pink	In a few months
Red	Soybean Specific

Figure 60. SoyKB to KBCommons tools status

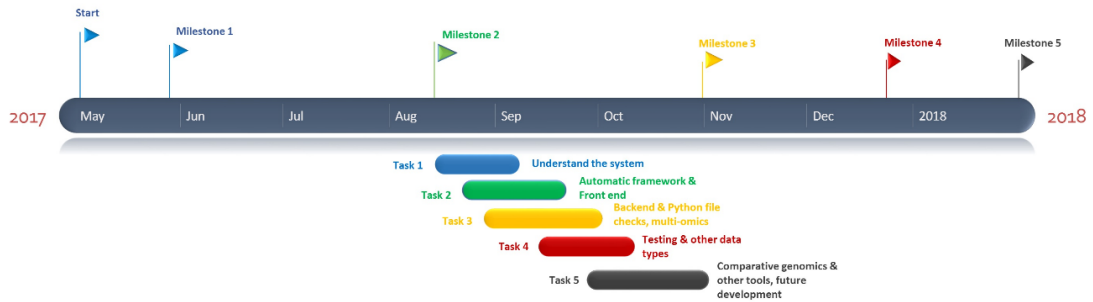


Figure 61. Timeline of the development

9. References

- [1] Joshi, T., Wang, J., Zhang, H., Chen, S., Zeng, S., Xu, B., & Xu, D. (2017). The Evolution of Soybean Knowledge Base (SoyKB). *Plant Genomics Databases*, 149-159. doi:10.1007/978-1-4939-6658-5_7
- [2] Joshi, T., Fitzpatrick, M. R., Chen, S., Liu, Y., Zhang, H., Endacott, R. Z., Gaudiello, E. C., Stacey, G., Nguyen, H. T., & Xu, D. (2014). Soybean knowledge base (SoyKB): a web resource for integration of soybean translational genomics and molecular breeding. *Nucleic Acids Research*, 42, 1245-1252. doi: 10.1093/nar/gkt905
- [3] Joshi, T., Patil, K., Fitzpatrick, M. R., Franklin, L. D., Yao, Q., Cook, J. R., Wang, Z., Libault, M., Brechenmacher, L., Valliyodan, B., Wu, X., Cheng, J., Stacey, G., Nguyen, H. T., & Xu, D. (2012). Soybean Knowledge Base (SoyKB): a web resource for soybean translational genomics. *BMC Genomics*, 13 Suppl 1:S15. doi: 10.1186/1471-2164-13-S1-S15
- [4] Reenskaug, T., & Coplien, J. (2009). The DCI Architecture: A New Vision of Object-Oriented Programming. Retrieved from http://www.artima.com/articles/dci_vision.html
- [5] Burbeck, S. (1992). Applications Programming in Smalltalk-80(TM): How to use ModelView-Controller (MVC). Retrieved from http://www.dgp.toronto.edu/~dwdgdr/teaching/csc2524/2012_F/papers/mvc.pdf
- [6] Davis, Ian. (2008). What Are The Benefits of MVC? *Internet Alchemy*. Retrieved from <http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/>

- [7] Surguy, M. (2013). History of Laravel PHP framework, Eloquence emerging. Retrieved from <https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/>
- [8] Bean, M. (2015). Laravel 5 Essentials: Explore the fundamentals of Laravel, one of the most expressive and robust PHP frameworks available. Available from <https://books.google.com/books?id=BWO4CAAQBAJ&printsec=frontcover#v=onepage&q&f=false>
- [9] Skvorc, B. (2015). Best PHP Framework for 2015 – SitePoint Survey Results. Retrieved from <https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/>
- [10] Otwell, T., et al. (2011). laravel/laravel: laravel/readme.md at master. Retrieved from <https://github.com/laravel/laravel>
- [11] Provos, N., Mazières, D., & Sutton, T. J. (1999). A Future-Adaptable Password Scheme. *Proceedings of 1999 USENIX Annual Technical Conference*: 81–92. Retrieved from https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node1.html
- [12] History of PHP. Retrieved from <https://secure.php.net/manual/en/history.php.php>
- [13] Summerfield, M. (2007). Rapid GUI Programming with Python and Qt: the definitive guide to pyqt programming. Available from <https://www.amazon.com/Rapid-GUI-Programming-Python-Definitive/dp/0134393333>
- [14] McConnell, S. (2009). Code Complete, Second Edition. Available from <https://www.vitalsource.com/products/code-complete-steve-mcconnell-v9780735636972>

- [15] About Python. Retrieved from <https://www.python.org/about/>
- [16] Hamano, J. C., et al. (2005). Releases – git. Retrieved from <https://github.com/git/git/releases>
- [17] Scopatz, A., & Huff, K. D. (2015). *Effective Computation in Physics*. *O'Reilly Media, Inc.* p. 351. Retrieved from <https://books.google.de/books?id=DYoNCgAAQBAJ&pg=PA351#v=onepage&q&f=false>
- [18] Bogard, J. (2012). Why GitHub's pricing model stinks (for us). *LosTechies*.
Archived at <https://web.archive.org/web/20150629153426/https://lostechies.com/jimmybogard/2012/11/07/why-githubs-pricing-model-stinks-for-us/>
- [19] The Problem With Putting All the World's Code in GitHub. (2015). *Wired*.
Archived at <https://web.archive.org/web/20150629152927/http://www.wired.com/2015/06/problem-putting-worlds-code-github/>
- [20] Gousios, G., Vasilescu, B., Serebrenik, A., & Zaidman, A. (2014). Lean GHTorrent: GitHub Data on Demand. Archived pdf at <http://www.win.tue.nl/~aserebre/msr14georgios.pdf>
- [21] Goodstein D. M., Shu S., Howson R., Neupane R., Hayes R. D., Fazo J., et al. (2012). Phytozome: a comparative platform for green plant genomics. *Nucleic Acids Res.* 40, D1178–D1186. doi: 10.1093/nar/gkr944

- [22] Yates, A., Akanni, W., Amode, M. R., Barrell, D., Billis, K., et al. (2016). Ensembl 2016, *Nucleic Acids Research*, 44, D710–D716. doi.org/10.1093/nar/gkv1157
- [23] Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., et al. (2013). GenBank, *Nucleic Acids Research*, 41, D36-42. doi: 10.1093/nar/gks1195
- [24] Mashima J., Kodama Y., Fujisawa T., Katayama T., Okuda Y., Kaminuma E., Ogasawara O., Okubo K., Nakamura Y., Takagi T. (2017). DNA Data Bank of Japan. *Nucleic Acids Research*, 45, D25-D31. doi: 10.1093/nar/gkw1001
- [25] Kaminuma E., Kosuge T., Kodama Y., et al. (2011). DDBJ progress report. *Nucleic Acids Research*, 39, D22–7. doi:10.1093/nar/gkq1041
- [26] Cochrane, G., Akhtar, R., Aldebert, P., Althorpe, N., Baldwin, A., Bates, K., Bhattacharyya, S., Bonfield, J., & Bower, L. (2007). Priorities for nucleotide trace, sequence and annotation data capture at the Ensembl Trace Archive and the EMBL Nucleotide Sequence Database. *Nucleic Acids Research*, 36, D5–D12. doi:10.1093/nar/gkm1018
- [27] Young, N. D., Debellé, F., Oldroyd, G. E., Geurts, R., Cannon, S. B., et al. (2011). The Medicago genome provides insight into the evolution of rhizobial symbioses. *Nature*, 480, 520-524. doi: 10.1038/nature10625.
- [28] Fields S., & Johnston M. (2005). Cell biology. Whither model organism research?. *Science*. 307 (5717): 1885–6. doi:10.1126/science.1108872.
- [29] Griffiths, E. C. (2010). What is a model? Archived at <http://www.emily-griffiths.postgrad.shef.ac.uk/models.pdf>

[30] Fox, M. A. (1986). *The Case for Animal Experimentation: An Evolutionary and Ethical Perspective*. *University of California Press*. Available from

[http://www.cell.com/trends/neurosciences/abstract/0166-2236\(87\)90036-1](http://www.cell.com/trends/neurosciences/abstract/0166-2236(87)90036-1)

[31] Rhodes, L. (2016). *Medicago truncatula*. The IUCN Red List of Threatened Species 2016: e.T176489A19401776

[32] Tang, H., Krishnakumar, V., Bidwell, S., Rosen, B., Chan, A., Zhou, S., Gentzbittel, L., Childs, K. L., Yandell, M., Gundlach, H., Mayer, K. F., Schwartz, D. C., & Town, C.D. (2014). An improved genome release (version Mt4.0) for the model legume *Medicago truncatula*. *BMC genomics*, 15:312. doi: 10.1186/1471-2164-15-312

VITA

My personal journey of coming from a small, remote village without having access to even the basic necessities such as schools and healthcare to the current day has been a journey full of challenges, opportunities, and most importantly great learning experiences. In our village, it is a rarity to finish high school especially for girl child who is often married off in her teens. With my persistence and dedication to studies, I was able to eventually overcome the many socio-economic challenges and excel academically.

During my high school, I realized that the only way for me to succeed in life is to go to college and pursue my passion for Mathematics and Science. In the high school board exams, I stood as an outstanding student by topping across our Mandal (county) and received a special award for this stunt. This boosted my confidence and brought me reputation in our village community.

With this support, encouragement, and interest I went to college to pursue Bachelor of Technology in Computer Science. Towards the completion of my Bachelors, I was in a confusion whether to pursue higher studies or to work in industry as I grew passionate about doing practical and impactful work. It was then I heard about Bioinformatics as a pioneering field at the intersection of biological, computational, and statistical research. I then decided to pursue Master of Technology in Bioinformatics.

To further gain hands on experience in research, I work at two bioinformatics research labs at the most premier research institutes in India: Indian Institute of Science and Indian Institute of Technology where I learned how crucial computational techniques are for scientific learning. With the research experience I gained, I am continuing research in Bioinformatics with Masters at the University of Missouri.