**Domain Identification in Alternative Spliced Genes**

**Introduction**follow these steps to run the website locally

download node js from(if not already installed):

<https://nodejs.org/en/>

download or clone the project from the DoChaP\_Shani repository

Afterwards, click on **downloadModules.sh** and wait for it to close.

If all has been downloaded in the past, click on **clickMe.sh** to use the website using chrome.

It can also be launched at this url: [http://localhost:3000](http://localhost:8000/index.html)

when finished go to cmd window and press **ctrl+c**.

**Server-Side Programming**

formed out of 2 modules besides the main running file. //right now

* Query Search Module

This module is responsible to find a gene record and its transcripts' information when a user searches via an input text field. Because we have different tables that contain important data which needs to be found by foreign IDs we decided to make several SQL queries. The query line should be filled with a gene name.

The module checks for matches In the gene\_symbol column of the Genes table.If no matches were found it would check different synonyms to find the closest match. There is a possibility were nothing is similar enough for the module to be associated with the query given. In this case, we would alert the user.

When a gene could be identified and its list of trancsripts, the function findTranscriptInfo would use each transcript record to find associated records of exons and domainEvents and their domainTypes. To clarify, domain Event are concrete cases of a domain type so general information can be found In a domain type record and specific information is in the domain event record which is why both records are needed.

Our main goal here is to return all of the information in a way that could be easily interpreted by our front-side modules. The results are in an javascript object/json format keeping an hierarchy of the data in this manner:

{

"gene":{*gene record*},

"transcripts":

[

*array of transcript following the syntax of*

{

*transcript record*,

"exons":[*array of exons records*],

"protein":

{

*protein record*,

"domains":[*array of domain*

*records*]

},

}

]

}

//an example for json will be displayed here when we will have an agreement on the syntax

List of queries used:

Getting record of the gene named – if exists.

SELECT \* FROM Genes WHERE gene\_symbol = '<geneName>'

Getting records of genes that have similar names to gene named

SELECT \* FROM Genes WHERE synonyms LIKE '%<geneName>%'

Getting all transcript records known for the gene. //we may need to check cases with 1 or 0 transcripts if its impossible

SELECT \* FROM Transcripts WHERE gene\_id = '<geneID>'

Getting all exon records for a specific transcript ID

SELECT \* FROM Exons WHERE transcript\_id = '<transcriptID>'

Getting a protein record for a specific transcript ID

SELECT \* FROM Proteins WHERE transcript\_id = '<transcriptID>'

Getting all domain records associated with a specific protein ID

SELECT \* FROM DomainEvent WHERE protein\_id = '<proteinID>'

Getting a domain type record that matches an domain type ID

SELECT \* FROM DomainType WHERE id = '<domainTypeID>'

* Database Utilities Module

This Module currently only opens a connection to the database when starting a server. Configurations will be made available from this module when the time comes. We hope to create an efficient module that can close and re-open the database only when needed. Debugging options and updating may be added to this module or in a separate module- depending on size and complexity.

DEBUG QUERIES:

Finding genes with no transcript (is it something we want?)

SELECT GeneID

FROM Genes

WHERE GeneID not in (SELECT gene\_id FROM Transcripts)

Finding transcript with no exons

SELECT refseq\_ID

FROM Transcripts

WHERE refseq\_ID not in (SELECT transcript\_id FROM Exons)

Finding transcripts with no protein

SELECT refseq\_ID

FROM Transcripts

WHERE protein\_id not in (SELECT refseq\_ID FROM Proteins)

Finding domain Type with domain Event

SELECT type\_id

FROM DomainEvent

WHERE type\_id not in (SELECT id FROM DomainType)

Finding mis-match between transcript and gene in protein records

SELECT refseq\_ID

FROM Proteins

WHERE gene\_id not in (SELECT gene\_id FROM Transcripts WHERE Proteins.transcript\_id=Transcripts.refseq\_ID)

To Do:

-check if tx area fully contains cds(is it something we want?)

-gene with 2 different yet identical transcript (is it something we want? Like gene 2218)

-each start is smaller then end(is it something we want? If so should we check s<e or s<=e)

-build class diagram to find more constraints

Things to look out for:

-A value for geneID that is for a different MGI-ID – a problem (which will we take)

- there is correct number of exons as in exon number

-a picture shows somewhere that the transcript id is an integer

-tx\_start<=cds\_start<cds\_end<=cds\_end?

-genomic\_start<genomic\_end?

-abs\_start<abs\_end?

-abs\_start<genomic\_start?

-gene where all transcript are the same size (could point to some problematic genes)

-all transcript with the same exon size (could point to a problem?or splices are not in cds?)

-domains with same genomic coordinates, length, gene but not type (could be duplicates)

- for each splice in domains exists the domain\_type, also for domain event

-nuc\_start<nuc\_end, aa\_start<aa\_end

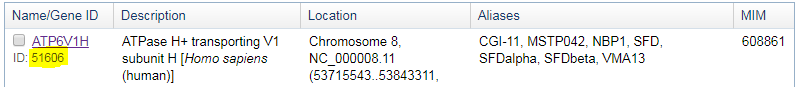
-type that is only in one transcript (could be duplicates)

-domain absolute positions are within absolute protein length

**Client-Side Programming**

Working with angularJS we created a website that now consists of 2 main pages. The first page is a searching page.

* search Module   
  We currently support searching only via a text field for a gene. The gene can be found by the following ways:
  + Its geneID as it shows in NCBI website. Example:



* + Gene symbol- a unique name given for a gene. Notice many organisms can have the same gene symbol but not the same geneID. Gene symbol is unique only per an organism. Because we only support *M\_musculus* it would be a problem. In the future when choosing to search by gene symbol you would have to select an organism. The gene symbol is case insensitive in this website.
  + Synonyms- we support many synonyms for each gene. A unique common official secondary name for a gene can be a synonyms and will also be case insensitive.

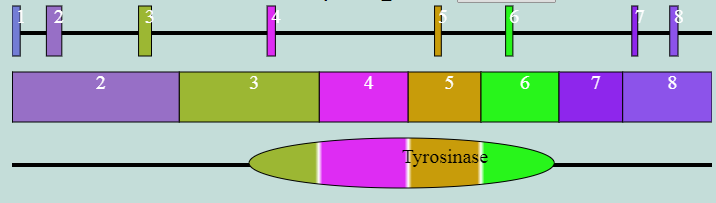
When making a search that might fit several genes the priority will be from first to last:

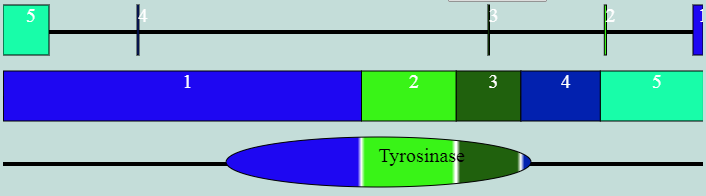
gene ID (case insensitive) > gene symbol(case insensitive) > synonym > (TO DO)similar gene symbol >similar synonym

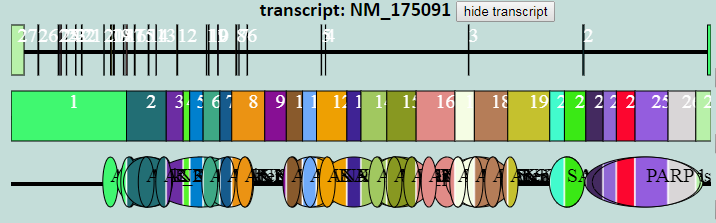
This module connects to our server. The results as they have been described in the Query Search Module will be delivered back to the client and the result page will load automatically.

* Result Module  
  the result module should print out analysis information regarding the wanted gene. When a gene is found information is gathered about it and its transcripts. The tool shows 3 views:
  + Transcript view(is it the correct name?): a line that imitates the gene's flow. Rectangles symbolize exons and plain lines introns (is it correct or can plain lines be UTR?) . This tool is helpful to see how the exons are spread and understand order and length proportions. Colors currently only help recognize the same exons in different views. The start and end are the coordinates of which the "earliest" transcript start and the "latest" transcript end respectively.
  + Exon view: exons from the transcript view are added together. To be exact, all of the exons shown are within the cds area and only exon parts which are coded into protein are shown. It is a description of how the cds is divided into exons since it shows the cds as chain of exons or part of exon if exons are not fully in the cds. Another way to look at it is a description of the structure of the protein by its exon origins. Notice that A situation where the first exons in the transcript (as shown in the transcript view) are not in the exon view if no parts of these exons are in the cds area.  
    Colors currently only help recognize the same exons in different views. This tool can also help the user understand order and length proportions as well as be helpful next to the domain view which will be described. The locations between transcript view and exon view cannot be compared, but location in the exon view and the protein view can be.
  + Protein view: the project focuses on domain structures in proteins. Therefore, the line symbolizes the protein structure and circles symbolize domains and where they are located based on the amino acid chain. Each domain is colored by the color/colors of the exons that are involved in the coding of the domain. The splice junctions in domain can be detected by white gradient line between exons. this can be useful when only a tip of the domain is in another exon and its exon color is not very visible due it being small. To be clear, each domain with more than one color is between exons and may be non-functional in a transcript where not all exons are present. The locations can be compared to the exon view since they are both describing the location in the amino acid chain of a protein.

Here are common struggles we had with the graphics: (to do: fix and explain why we chose each solution

* + Exons are so small (in proportion to the cds area) it can barely be clicked or can be filled with text. Also introns are very big 
  + Because of the strands the location is backwards and the exons are numbered correctly by the order in the protein but not in transcript view so it is confusing



* + Domain areas overlap  
    
  + Great number of exons/ long proteins. 11% percent of current transcripts have more than 20 exons. The example shows the only transcript with more than 300 exons  
    