BLOSSOMSTTM HUB – AN ONLINE TOOL FOR DESIGNING STTM VECTORS AND VISUALIZING PHENOTYPIC CHANGES OF STTM TRANSGENIC LINES

Avinash Subramanian
Michigan Technological University, avsubram@mtu.edu

BLOSSOM STTM HUB – AN ONLINE TOOL FOR DESIGNING STTM VECTORS AND VISUALIZING PHENOTYPIC CHANGES OF STTM TRANSGENIC LINES

By

Avinash Subramanian

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2017

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Computer Science

Report Advisor:	*Dr. Hairong Wei*
Committee Member:	*Dr. Zhenlin Wang*
Committee Member:	*Dr. Myounghoon Jeon*
Department Chair:	*Dr. Min Song*

Contents

List of Figures

Abstract

Small RNAs including microRNAs (miRNAs) and short interfering RNAs (siRNAs) are widely present in plants. They are transcribed from non-coding small RNA genes and then play as regulators to modulate the levels of messenger RNAs (mRNAs) of protein-coding genes via sequence pairings. This is because a paired complementary double sequence helix structure can trigger mRNA degradation or interfere with mRNA translation. Short Tandem Target Mimic (STTM) is a recently developed technology that can be used to produce a complementary sequence to a miRNA and destroy it or reduce the expression level of this miRNA via the formation of paired double-strand structure. Research has shown success in plant species like *Arabidopsis* [1] and tomato [2]. The main motivation of this thesis report is to describe a web application named DesignSTTM portal, which has been developed to computerize and automate the design of a STTM sequence that enables one to target a given miRNA species for degradation via sequence complementarities mechanism. The DesignSTTM then inserts the designed STTM sequence immediately downstream of the Cauliflower mosaic virus promoter called 35S in a DNA plasmid vector that can carry 35S promoter + STTM expression cassette into nuclei, and then insert it into a genome, where 35S can drive a STTM sequence to express and produce complementary sequences to target miRNA for silencing. Before the DesignSTTM portal displays the designed plasmid vector carrying STTM sequence in the form of a circular map, it uses an algorithm called basic local alignment search tool (BLAST) to query a locally installed vector database and then annotate all the elements in the plasmid vector. The elements that will be identified and annotated include all the genes, restriction enzyme sites, promoters, and the location of the inserted STTM sequence, which will then be plotted as features of plasmid vector in a map. The portal also displays the plasmid DNA sequence resulting after the insertion of STTM sequence, with a highlight of the location of the STTM sequence. The DesignSTTM portal also has a function to design the primer sequences used to amplify the STTM sequence and majority of plasmid vector using Polymerase Chain Reaction (PCR) technology and transfer them to a binary vector. The DesignSTTM portal is also capable of producing a PDF document incorporating all the derived results, and sending the file as an attachment to users' email. Blossom STTM Hub's also maintains an account for each user to store all the obtained results. The other web portal that was implemented in this Blossom STTM Hub's is called MaterialSTTM, which was designed to store and visualize both genotypic and phenotypic data collected from STTM transgenic lines generated through transformation of STTM plasmid vectors into different plant species. Users can purchase the transgenic seeds available on MaterialSTTM portal by placing an order through the e-commerce store set up through the MTU Touchnet service. This tool will be instrumental for numerous plant biologists who study the functions of miRNAs species through modulating their expression levels in plants.

1 Introduction

1.1 MicroRNAs and Their Functions in Plants

MicroRNAs (miRNAs) are recently discovered non-coding small RNAs with a length of around 20 to 24 nucleotides [3]. The major function of a miRNA in an organism is to interfere with a target message RNA (mRNA) through forming paired double-strand structure that leads to the reduction of mRNA level or suppression of its translation. Therefore, miRNAs play a critical role in modulating mRNA levels in plant. Since abundances of mRNA species represent the levels of gene expression, researchers usually consider miRNAs as regulators in turning down gene expression levels. Although they are conserved in both plants, plant miRNAs interfere with their target mRNAs primarily through perfect or nearly perfect sequence pairings. The paired double-strand structures can trigger a series of events that lead to either the degradation of target mRNAs through effective cleavage or the suppression of mRNA translation. This has laid the foundation for using various molecular techniques to characterize the functions of plant miRNAs.

1.2 Short Tandem Target Mimic (STTM)

STTM is a new technology developed by Dr. Guiliang Tang's laboratory recently [4] that performs a very special task: modulating the expression level of one or two miRNAs in plants [4]. Each STTM vector carrying a STTM structure is designed based on one or two target miRNAs nucleotide sequences. It can target the specific miRNA(s) for destruction without disturbing the other miRNAs present in the species. This is a major advancement in understanding miRNA functions in major crops like rice, *Arabidopsis*, maize and soybeans. This technology enables us to learn miRNA functions through a loss-of-function mechanism.

A STTM structure can destroy one or two targeted miRNAs through sequence pairing. A typical STTM sequence is designed to generate the RNA sequence complementary to one or two target miRNAs. When a STTM structure is expressed, it produces one or two single strand small RNA molecules that are complementary to target mRNAs, leading to the formation of the double stranded RNA(s) between STTM RNA sequence and its target miRNA(s), which then trigger(s) a series of scenarios that leads to the degradation of STTM targeted miRNAs, allowing for probing and characterization of miRNA functions.

1.3 Objectives and Goals of Blossom STTM Hub

Designing STTM manually is time-consuming and error-prone. To reduce the workload for biologists, we developed Blossom STTM Hub. The objectives of the Blossom STTM Hub include the following:

1.3.1 To Develop a Web Portal Called DesignSTTM to Automate STTM Vector Design

Dr. Guiliang’s laboratory at Michigan Technological University published the STTM vector design method. The tool’s algorithm should mimic the manual design process and transform that labor-intensive design process into a computational construction process. For biologists and researchers around the world, we attempt to provide a tool that implements Dr. Guiliang’s method to design STTM vector sequences and deliver accurate and annotated plasmid maps of the designed vectors. This portal facilitates utilization of the published STTM method and inculcates further growth around this field of research.

1.3.2 To Develop a Web Portal Called MaterialSTTM to Display the Phenotypic and Genotypic Data Generated from STTM Transgenic Lines

The MaterialSTTM is designed to store the designed STTM vectors and the phonotypical genotypic data of STTM transgenic lines online so that other researchers in the plant research community can query and study the functions of miRNA species. Specifically, the system is to display the transgenic lines showing genotypic and phenotypic changes after the target miRNAs being destroyed by deigned STTM structures.

1.3.3 To Develop a Genome Browser to Visualize Genome-Wide Expression Data Generated from STTM Transgenic Lines

A genome browser is developed to display various RNA-seq and ribo-seq data produced from transgenic lines of various miRNAs-STTM structures, this web portal is still under construction.

2 Background Study

After defining the research objectives, I searched and studied the available tools and techniques that are potentially useful to this web application, especially those that are of similar applications. Understanding the underlying architectures and methods of operation in similar applications helps to develop an efficient web application. The following are the web applications that were explored:

2.1 PlasMapper

PlasMapper [5] is an online tool developed by David Wishart et al., and it can generate annotated high-quality circular plasmid maps with input DNA vector sequences. It makes use of the Basic Local Alignment Search Tool (BLAST) [6] to annotate the genes and other features within the plasmid, map all the restriction enzyme digestion sites, and then labels them accurately. It has the capability to detect promoters, terminators, cloning sites, open reading frames, and replication origins present in the input DNA vector sequences. The generated plasmid maps are shown as a graph with high resolution. It also provides various customizable options that users can utilize to get a clearer picture of a DNA vector. The output can also be obtained in different image formats as needed. The PlasMapper uses Java and C as a combination to prepare the plasmid maps.

2.2 SnapGene Viewer

SnapGene Viewer is powerful software that also computes and draws circular plasmid maps of DNA vectors and has a wide range of features from which users can choose. It detects all the restriction enzyme digestion sites and other features, and provides several dynamic functionalities to add and customize vectors. The main drawback of the SnapGene Viewer could be that it is not an online tool, and users have to download it to use Snapgene as a stand-alone software application. Blossom STTM Hub needs to function online, and it does not require some of the advanced functionalities of the SnapGene Viewer.

2.3 The Technologies Available to Be Used

After analyzing the objectives and requirements of the Blossom STTM Hub online tool project, it became clear that it needed to function as a web application with spectacular speed, security, efficiency, and accuracy. The Blossom STTM Hub is based on the Michigan Technological University Linux operated Apache server named “blossom.ffr.mtu.edu,” and the server has MySQL Database Service installed in it. We decided to proceed with PHP for server side scripting to complete the LAMP platform, which works well for dynamic web applications [7]. The LAMP platform consists of four components that are structured in a layered way and all four components form integral parts of the software application. The layers of the LAMP platform are collectively known as web stack when used for a standalone web application. Linux – It is the lowest level layer and acts as the operating system that has the ability to run all the remaining layers of the software stack. The next layer in the LAMP platform is Apache which is a web server that provide the mechanics to render a web page on the Internet. MySQL provides the data storage layer of the LAMP platform and uses relational databases to store large and complex data that can be accessed through SQL queries. The last layer in the LAMP platform is PHP and it acts as a binding layer to other layers. PHP resides inside the Apache web server and is used to create dynamic content capable of using the data stored in the MySQL database. To communicate with server, a server side scripting technique needs to be used. A few examples of server side scripting languages are JSP (Java Server Pages), Perl CGI, and PHP. The main difference between PHP as compared to JSP and other scripting languages is that PHP code is interpreted by the server directly whereas JSP codes and others are executed by a virtual machine. For large scale multipurpose applications, using a virtual machine has a positive impact but for the requirements of our portal, interpreted server code is the least computationally intensive technique and provides better performance. This is the main reason why PHP is chosen to be used as the serve side scripting language over other alternatives.

```
<html>….<p>The current system date is <?php echo date(); ?></p> ….</html>
```

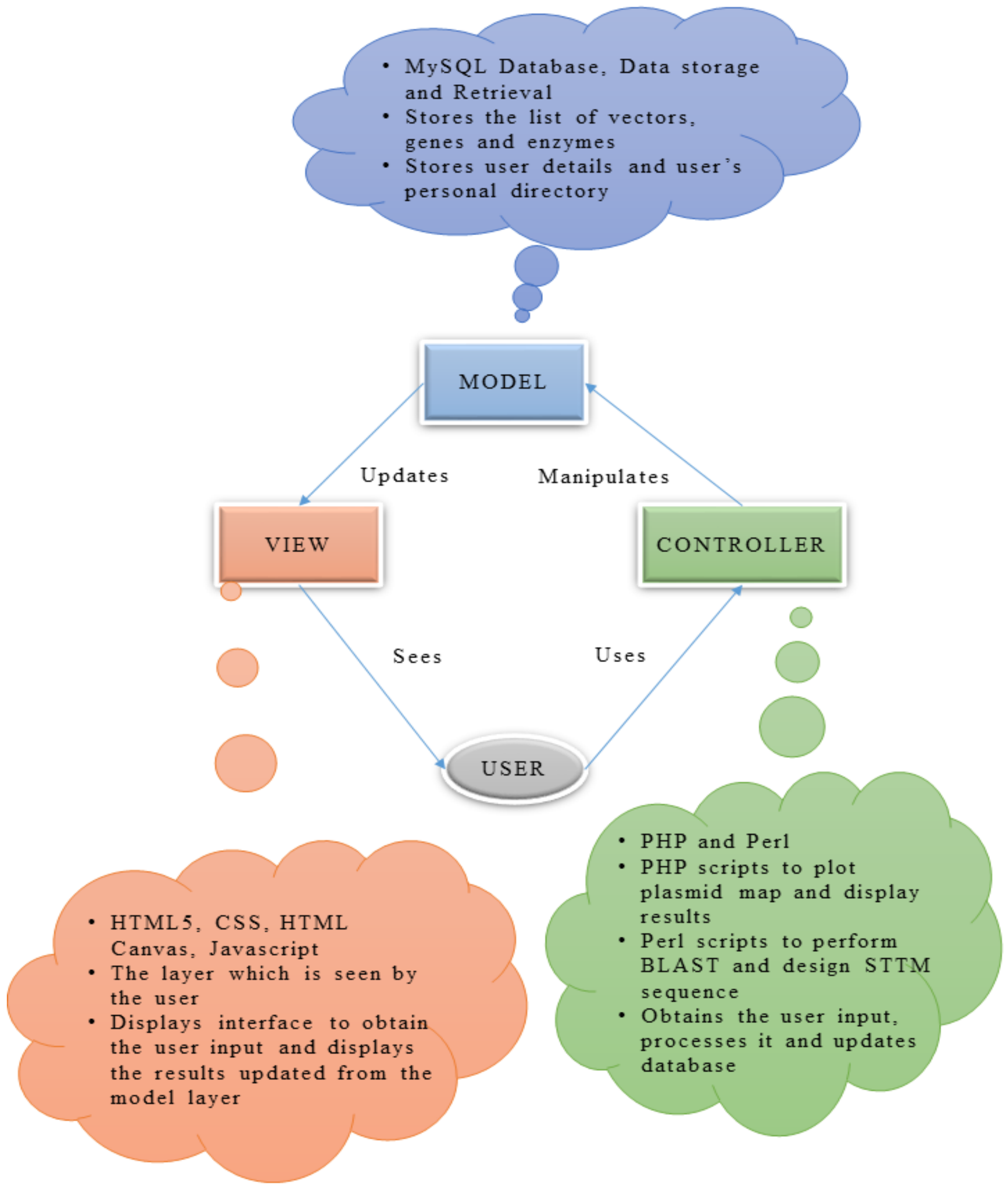
The above code shows how a PHP code is embedded into a HTML program. To perform this using Java would require the HTML to redirect to a Java scriptlet and retrieve the output back after the Java Virtual Machine processes the result. Java functions are faster and more powerful for complex server side computations, but for a Web portal such as the DesignSTTM in which numerous client side computations take place, the PHP-HTML embedded capabilities make it a better alternative [8]. This makes the overall system less dependent on the server for small errands and less error prone. The tool also needs to use Perl to call executable software and BLAST to annotate the DNA vectors. For the front-end development, CSS, HTML5, and HTML Canvas have been used. HTML5 [9] was chosen because of its state of the art features and core simplicity. The HTML Canvas 2D Context API provides an empty canvas (a blank bitmap) to draw shapes, text, or images on it. Canvas content can be easily modified applying transforms (scale, rotate, or translate), compositing coordinate changes, or modifying shadow attributes. It offers good performance in complex scenes and features like dumping the content of the canvas to an image element [10]. HTML5 Canvas can be used to plot drawings automatically [11], and it turns out to be faster than other available languages like Java Applets or Swing [12]. JavaScript has been used for computing the pixel locations while plotting the DNA plasmid map, and for performing client side validations and JQuery for implementing certain dynamic functionalities like automatic scrolling, multi-page parsing, and background processing [13].

3 Design

As aforementioned, this Web application contains two Web portals, DesignSTTM and MaterialSTTM, and one genome browser for visualizing RNA-seq or ribo-seq data yielded from STTM transgenic lines. The DesignSTTM and MaterialSTTM Web portals are based on the MVC architecture.

3.1 Model View Controller Architecture

The DesignSTTM Web portal was built on the Model View Controller (MVC) architecture. The Model layer contains all users' information, enzyme digestion sites, list of vectors and their sequences, and information about users' directories. All these are stored as tables in a MySQL Database. Then, the View layer that forms the front-end user interface for the Web portal is developed using HTML5, CSS, and Canvas programming languages. The View layer is what a user sees. It consists of the interface through which the user provides input and views the final output updated by the Model layer. The user does not have access to the Model layer directly. This provides the application with data encapsulation feature. The final layer is the Controller layer, which acts as a bridge between the user and the Model layer. The functions of controller layer are to obtain the user input, process them, and manipulate the Model layer. In our Web portal, the Controller layer uses PHP and Perl scripts. Designindex.php is the PHP script that takes the input obtained to the Model layer, and Index3copy2.php is the PHP script that uses the input to plot the plasmid map and other results of the DNA vector containing the STTM sequence. BLAST.pl and miRNA.pl are the Perl scripts that implement the BLAST algorithm for annotation of DNA vectors and designing the STTM sequence and the primers, respectively.



• MySQL Database, Data storage and Retrieval
• Stores the list of vectors, genes and enzymes
• Stores user details and user's personal directory
MODEL
Updates
Manipulates
VIEW
CONTROLLER
Sees
Uses
USER
• PHP and Perl
• PHP scripts to plot plasmid map and display results
• Perl scripts to perform BLAST and design STTM sequence
• Obtains the user input, processes it and updates database
• HTML5, CSS, HTML Canvas, Javascript
• The layer which is seen by the user
• Displays interface to obtain the user input and displays the results updated from the model layer

Figure 3.1: Depicts the Model, View, and Controller Layered Architectural pattern followed by the Blossom STTM Web portal.

3.2 Development of DesignSTTM Web Portal

The workflow of DesignSTTM explains the multiple steps in designing a STTM vector starting from an input miRNA sequence. First, a user enters the miRNA sequence and its name. The entered sequence is checked for validity—the miRNA sequence has to be between 18 and 30 bp (bp - base pair is a unit consisting of two nucleobases bound to each other by hydrogen bonds. They form the building blocks of the DNA double helix, and contribute to the folded structure of both DNA and RNA. Here bp acts as a unit of measure for the length of sequences) in length and must contain only the characters "atgc" or "augc." If the input is invalid, the system alerts the user. Then the user needs to select the vector sequences to include the designed STTM sequence (like pOT2). Next, the user has the option of transferring the designed STTM sequence into a binary vector. If the user chooses to transfer STTM structure into a binary vector, the user needs to identify one or two restriction enzyme sites to insert the STTM sequence structure into the binary vector (also called destination vectors). Then the index3copy2 program is called to execute to prepare the STTM vector sequence. If the user does not wish to transfer the STTM sequence to a binary vector, the STTM sequence will be inserted into an intermediate vector like pOT2 when the index2copy3 program is called for execution.

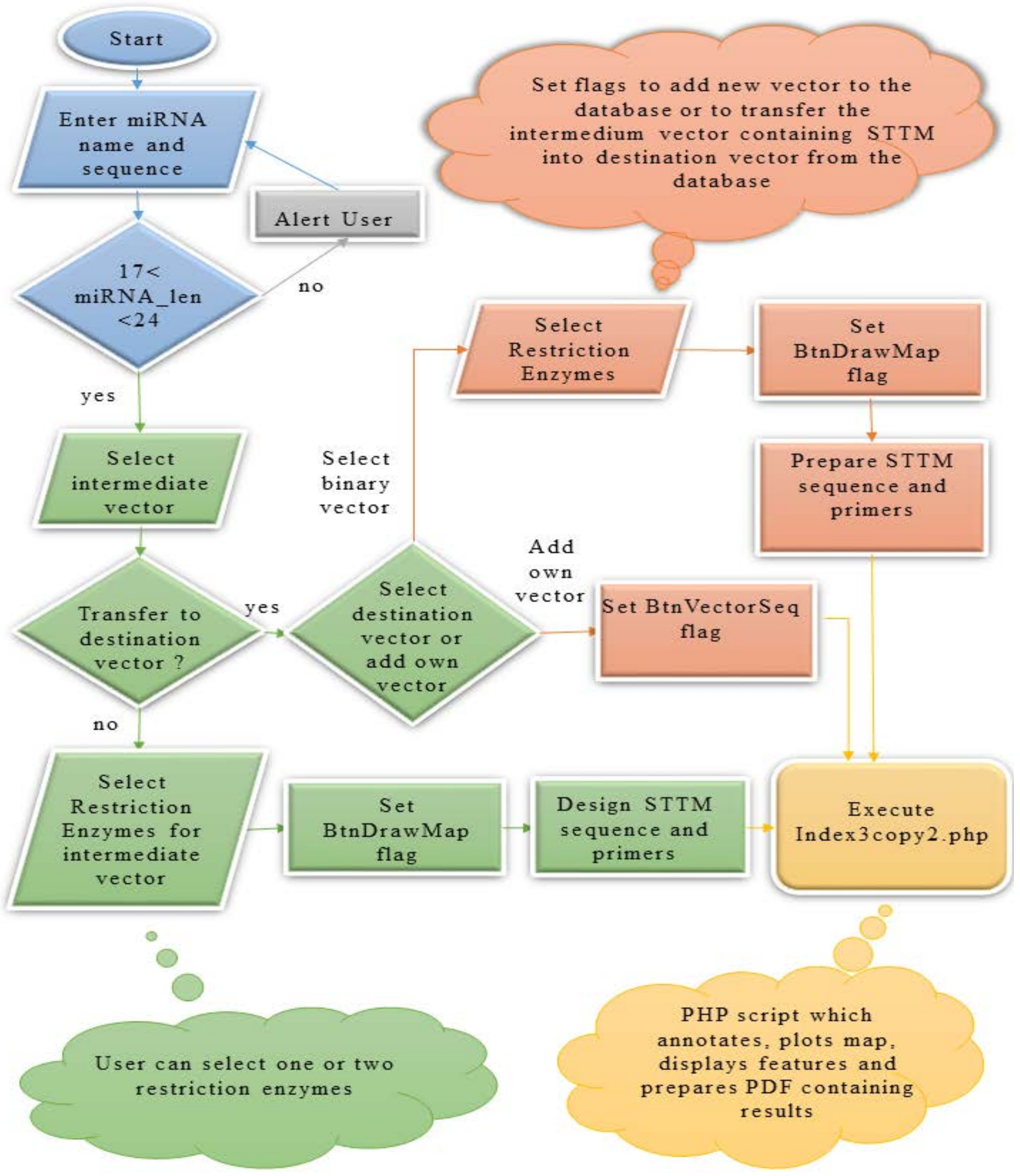

Start
Enter miRNA name and sequence
Alert User
17< miRNA_len <24
no
yes
Select intermediate vector
Transfer to destination vector ?
yes
no
Select destination vector or add own vector
Select binary vector
Add own vector
Set flags to add new vector to the database or to transfer the intermedium vector containing STTM into destination vector from the database
Select Restriction Enzymes
Set BtnDrawMap flag
Prepare STTM sequence and primers
Set BtnVectorSeq flag
Select Restriction Enzymes for intermediate vector
Set BtnDrawMap flag
Design STTM sequence and primers
Execute Index3copy2.php
User can select one or two restriction enzymes
PHP script which annotates, plots map, displays features and prepares PDF containing results

Figure 3.2: The workflow of DesignSTTM depicting the design process of STTM vector. The designed STTM structure was inserted into an intermediate vector, and then transferred to a binary vector that can be delivered into plant cells via Agrobacterium.

3.3 DesignSTTM Result

The DesignSTTM-index3copy2 PHP script obtains the designed STTM sequence and inserts it into a vector uploaded by the user to produce the miRNA targeted STTM Plasmid Vector, which is displayed as a map on the Web portal. The system also uses the script when the user uploads a new vector to the Web portal. In this case, the script will use the BLAST algorithm to plot the features present in the uploaded vector and send the control back to Design STTM portal, where the user will now be able to use the vector they have uploaded. The following steps occur if the user passes the destination vector and STTM sequence to the index3copy2 script:

- Find the position of restriction enzymes in the destination vector.
- Insert the designed STTM sequence from the previous step into the destination vector at one the restriction enzyme site or between two restriction enzyme sites and use the BLAST algorithm to annotate the features and gene families present in the miRNA targeted STTM destination vector.
- The circular plasmid map is then plotted using HTML5 Canvas, and the annotated enzymes are labeled across the plasmid map using Draw_Arc() and Draw_Enzyme_Label() functions, which will be discussed in a later section.

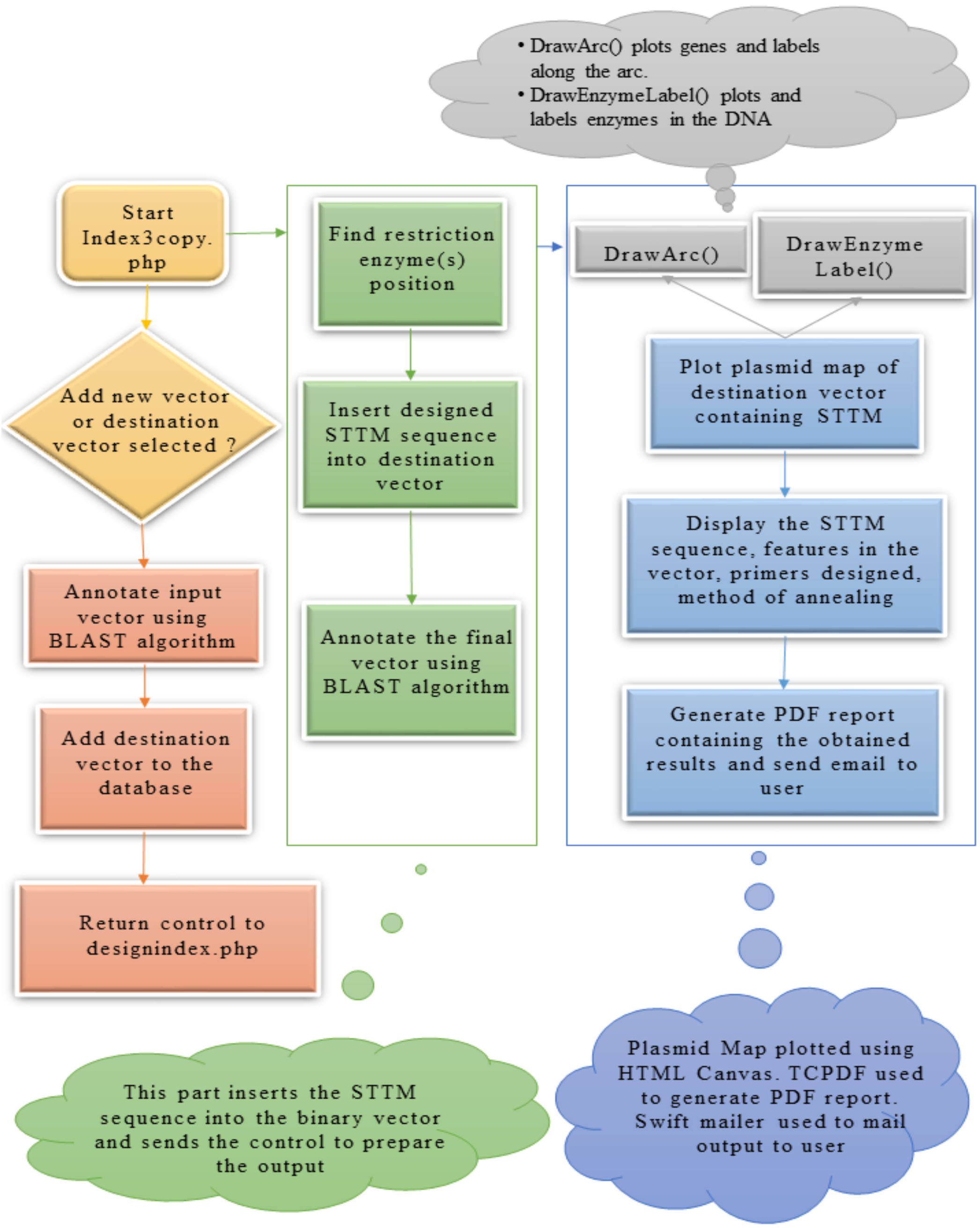

• DrawArc() plots genes and labels along the arc.
• DrawEnzymeLabel() plots and labels enzymes in the DNA
Start Index3copy.php
Add new vector or destination vector selected ?
Annotate input vector using BLAST algorithm
Add destination vector to the database
Return control to designindex.php
Find restriction enzyme(s) position
Insert designed STTM sequence into destination vector
Annotate the final vector using BLAST algorithm
DrawArc()
DrawEnzyme Label()
Plot plasmid map of destination vector containing STTM
Display the STTM sequence, features in the vector, primers designed, method of annealing
Generate PDF report containing the obtained results and send email to user
This part inserts the STTM sequence into the binary vector and sends the control to prepare the output
Plasmid Map plotted using HTML Canvas. TCPDF used to generate PDF report. Swift mailer used to mail output to user

Figure 3.3: Design process of DesignSTTM module encoded in index3copy2 program.

3.4 Database Schema

The Blossom STTM Web portal uses a MySQL Database. The database structure implemented in the Web portal is a simple single database instance called STTM. There are five distinct tables used in the STTM Database. The “userdetails” table stores a user’s profile information including username, encrypted password, email, and phone number. The system will use this information to verify the user during login and to identify the user when generating plasmid maps. The “root_vector” table contains the vector name, enzyme name, and the starting position storing the features for the default vectors available for the user to use (Ex. p5941). The “uservectors” table is a table created for every user to add his own annotated vectors to the database. The “enzyme_list” is the table that contains the selective list of available restriction enzymes shortlisted from the REBASE NEB database. The schema also has the highlighted “sttm_destination_vectors” table that will be implemented in the future and this table will store all users’ resultant destination vectors after insertion of STTM sequence. The “transgenic” table stores the primers and species names of the various transgenic lines expressed in the STTM MaterialSTTM Web portal. The “phenotypic” table stores the image reference names for each of the phenotypic changes shown for the STTM transgenic line of the species. The “admin_userdetails” table stores the user details who will act as admins and will be populating the “transgenic” and “phenotypic” tables from the portal.

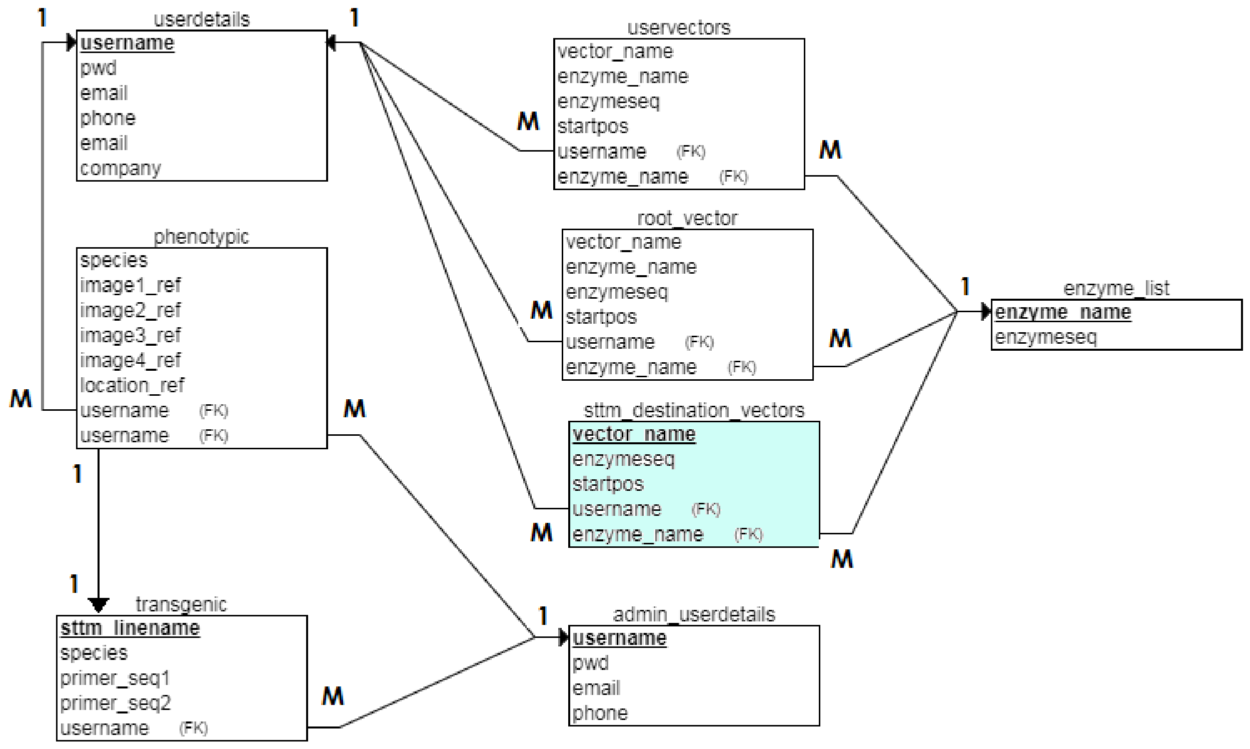


Figure 3.4: Database schema for Blossom STTM Hub.

4 Algorithm to Plot Plasmid Map

To draw the designed STTM constructs and display the features are a very difficult task. The challenges lie in three aspects:

1) We need to identify the features in a designed STTM construct.
2) We need to illustrate and label the features precisely along the circular DNA.
3) We need to label the restriction enzyme digestion sites along the circular DNA.

To overcome all the above challenge, a custom solution needs to be developed that would work for plasmids of different lengths (plasmids can be of drastically different lengths—they can be less than 4000kb or even greater than 15000kb). The circular plasmid map should be consistent in appearance, and all the enzymes should be clearly labeled without any overlaps.

4.1 Construction of STTM Binary Vector

As mentioned in the previous section, the DesignSTTM Web Portal designs a STTM sequence for each miRNA, inserts it into the destination binary vector, annotates the vector using BLAST.pl, and plots the plasmid DNA map along with other results. The circular map of the plasmid vector DNA containing the STTM construct is plotted using HTML Canvas. The Canvas element has a defined width and height that is set dynamically based on the size of the plasmid. The circular plasmid in plotted on the canvas with all genes, promoters, terminators and restriction enzymes along with the designed STTM sequence marked and labeled. Then the features of the plasmid are plotted on the canvas by translating the starting and ending locations of all the features to X & Y pixel coordinates with scale. On completion of this process, the system converts the canvas element to an image, displaying it on the screen. The following Algorithms have been designed to plot the plasmid map using HTML <canvas> element that renders graphics on a webpage:

4.1.1 Blossom Plasmid Map Plotting Algorithm

The Blossom Plasmid Map Plotting Algorithm plots the destination binary vector as a circular map. The algorithm receives the input from the BLAST results of the destination binary vector and performs recursive calls to the function DRAW_ARC() for every identified feature (enzymes, genes, promoters, and terminators). Comparing to the available radius on the Canvas to the length of the destination binary vector, the algorithm determines the scale to which the plasmid can be plotted. The circumference of the circular plasmid map to be plotted is divided into as many parts according to the length of the plasmid. Now the DRAW_ARC() function plots each feature from the BLAST result on the divided circular plasmid map. Refer to Appendix A for the code that runs draw_arc() function.

Function: DRAW_ARC()

function draw_arc(start, end, lnwidth, color, label_x_val, label_y_val, is_direction, comp)

Parameters and definitions:

Start → the start position of a feature

End → the end position of a feature

Lnwidth → the width of the line to be drawn

Color → the color chosen from a pool of colors

Label_x_val → The current X coordinate header position to label the feature

Label_y_val → The current y coordinate header position to label the feature

Is_direction → represents the orientation of the feature obtained from the Blast program

Comp → The name of the feature

Note: Here feature refers to any of the enzyme, promoter, terminator or origin

Algorithm for DRAW_ARC()

```
A. Begin procedure draw_arc(start, end, lnwidth, color, label_x_val, label_y_val, is_direction, comp)
B. radius = RADIUS_OF_CIRCULAR_PLASMID
C. counter = comp[].length
```

```
For I = 0 to counter
    temp1 = ( start / total_nucleotides ) * ( 2 * π * radius)
    temp2 = ( end / total_nucleotides ) * ( 2 * π * radius)
    //Draw Arc from temp1 to temp2 using context.arc()
    Context.arc(radius, temp1, temp2)
    context.lineWidth = lnwidth
    context.strokeStyle = color

    if is_direction == PLUS
        arrow1_startx = (label_x_val + 15 pixels)
        arrow1_starty = (label_y_val - 15 pixels)
        arrow2_startx = (label_x_val - 15 pixels)
        arrow2_starty = (label_y_val - 15 pixels)
        arrow3_startx = (label_x_val + 15 pixels)
        arrow3_starty = (label_y_val + 15 pixels)
    end if

    if is_direction == MINUS
        arrow3_startx = (label_x_val)
        arrow3_starty = (label_y_val)
        arrow2_startx = (label_x_val + 15 pixels)
        arrow2_starty = (label_y_val - 15 pixels)
```

```
            arrow3_startx = (label_x_val - 15 pixels)
            arrow3_starty = (label_y_val - 15 pixels)
        end if
        // Form a triangle with the three points
        Move to (arrow1_startx, arrow1_starty)
        Line To (arrow2_startx, arrow2_starty)
        Move to (arrow2_startx, arrow2_starty)
        Line To (arrow3_startx, arrow3_starty)
        Move to (arrow3_startx, arrow3_starty)
        Line To (arrow1_startx, arrow1_starty)
    end for
D. End procedure
```

4.1.2 Blossom Plasmid Large Features Labeling Algorithm

After plotting of the BLAST results on the canvas to form the circular plasmid, it could not be interpreted by a user if the genes and enzymes are not labeled. This was a more significant practical challenge than plotting the features. The main challenges are as follows:

- In a region of the plasmid where many enzymes need to be labeled, using an algorithm to label enzymes without getting overlapped was a major challenge.
- There was a need to differentiate between labeling enzymes and other gene features in order to identify them clearly.

In order to overcome the above difficulties, I came up with two algorithms—one to label large gene features using curved text along circular plasmid's inner circle and the other algorithm to label the smaller enzymes features with label lines from the outer circle of the circular plasmid:

function drawTextAlongArc(context, str, centerX, centerY, radius, start, end)

Parameters and definitions:

Context → the current focus header (focus header is the flat Cartesian surface whose origin (0,0) is at the top left corner where the <canvas> begins and its x-coordinate increases moving right and y-coordinate increases when going down the canvas.)

Str → the string name of the feature to be labeled

centerX → the X coordinate of the center position of the feature

center → The Y coordinate of the center position of the feature

radius → The radius of the canvas circle

start → starting position of the feature

end → ending position of the feature

Note: Here feature refers to any of the enzyme, promoter, terminator or origin

Refer to Appendix B for the code in drawTextAlongArc()

Algorithm of DrawTextAlongArc()

```
Begin procedure drawTextAlongArc(context, str, centerX, centerY, radius, start, end)
B. totalbp = total number of nucleotides in vector
C. radius = RADIUS_OF_CIRCULAR_PLASMID
D. if ( start < ( totalbp ) * 0.25 ) || (start > (totalbp)*0.75)
        multiplier = ( end / totalbp ) * 2 * π * radius
```

```
      angle = (-1) * 2 * $\pi$ * ( ( end – start ) / totalbp )
      len = string length of (str[])
      Rotate cursor by multiplier
      for i = 0 to len
          Rotate cursor by (-1)*(angle/len)/2
          currentFocus = cursor position
          context.fillText( str [ i ] )
          Restore cursor position to currentFocus
      end for
  end if
E. if ( start > ( totalbp ) * 0.25 ) AND ( start < ( totalbp ) * 0.75)
      multiplier = ( start / totalbp ) * 2 * $\pi$ * radius
      angle = (+1) * 2 * $\pi$ * ( ( end – start ) / totalbp )
      len = string length of (str)
      Rotate cursor by multiplier
      for i = 0 to len
          Rotate cursor by (-1) * ( angle / len ) / 2
          currentFocus = cursor position
          context.fillText( str [ i ] )
          Restore cursor position to currentFocus
      end for
  end if
```

F. End procedure

The above algorithm was developed to plot the principal features that are at least one-twentieth as long as the size of the destination binary vector, and the remaining features and restriction enzyme digestion sites are marked and labeled outside the circular rings. The restriction enzyme digestion sites and smaller features are plotted and labeled using draw_enzyme() function. The reason the threshold length for the features along the curve to be fixed at least one-twentieth was to avoid potential overlapping of labels.

4.2 Blossom Plasmid Octant Divided Small Features and Restriction Enzyme Sites Labeling Algorithm

DNA vectors usually contain many features like restriction enzyme digestion sites, promoters, gene coding regions, terminators, plasmid vector replication origin segments, and so on. Annotating and labeling of the designed plasmid circular maps is necessary because so many enzyme digestion sites often become crowded around the same regions. As such, there is a high chance that the labels of the different enzyme sites may overlap with each other. A foolproof algorithm was designed for labeling enzyme site specifically to comply with the plasmid map generated by the DesignSTTM Web portal. Figure 4.1 shows the concept of splitting the circular plasmid into eight octants for labeling. Figure 4.2 shows a rough example of labeling the enzymes in an octant.

In order to obtain the list of gene families and features present in the vector sequence, we used the NCBI tool called the BLAST (Basic Local Alignment Search Tool) [24]. The BLAST finds regions of similarity between biological sequences, and the BLAST finds regions of local similarity between sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches of similar gene families present in the vector. We have used the curtailed “emvec” database containing vector sequences to perform the BLAST for our plasmid vectors.

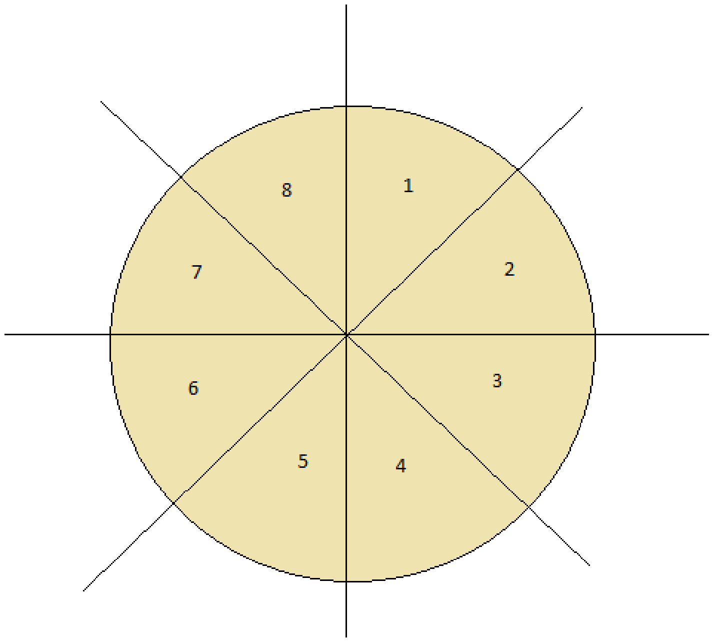


Figure 4.1: Circular plasmid divided into octants to facilitate labeling of restriction enzyme sites in the binary vector without overlapping text labels

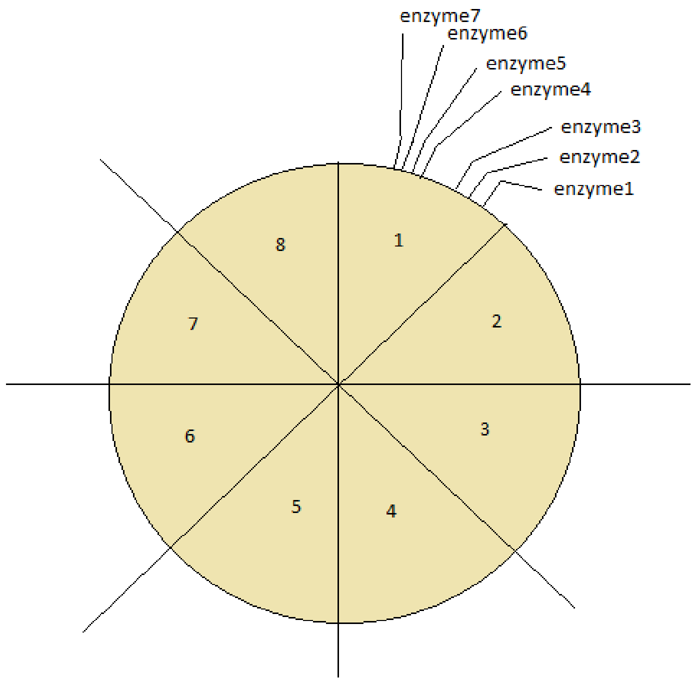


Figure 4.2: Algorithm used to label restriction enzyme sites inside every octant.

Function draw_enzyme():

draw_enzyme(start,end,lnwidth,color,label_x_val,label_y_val,is_direction, comp ,isgene)

Parameters and definitions:

Start → starting position of the enzyme

End → ending position of the enzyme

Lnwidth → width of the line to be drawn to label the enzyme

Color → color of the labeling text (red if it is a small gene or STTM sequence, black otherwise)

Label_x_val → X coordinate of the enzyme label position

Label_y_val → Y coordinate of the enzyme label position

Is_direction → orientation of the enzyme as obtained from BLAST program

Comp → name of the enzyme to be labeled

Isgene → the flag which determines if the feature is an enzyme or a small gene

Note: Here small gene refers to features of the vector DNA that are lesser than 5% of the total size of the vector DNA.

Refer to Appendix C for the code in draw_enzyme() function

Algorithm for draw_enzyme():

```
A. Begin procedure
draw_enzyme (start,end,lnwidth,color,label_x_val,label_y_val,is_direction,comp,isgene)
B. radius = RADIUS_OF_CIRCULAR_PLASMID
C. startbp = ( start * totalbp ) / ( 2 * π )
D. endbp = ( end * totalbp ) / ( 2 * π )
E. if "STTM_" ⊂ comp            // Check if comp has the substring STTM_
        context.strokeStyle = "brown"
   else if isgene == 1
        context.strokeStyle = "red"
   else
        context.strokeStyle = "black"
   end if
F. Draw fixed short line from enzyme position
   context.moveTo(label_x_val, label_y_val)
   context.lineTo(label_x_val+15, label_y_val+15)
G. line1_startx = label_x_val+15
```

```
   line1_starty = label_y_val+15
H. angle = (startbp + endbp)/2
I. OCTANT1CONDITION = ((angle > (-0.50) * (π)) && (angle < ((-0.25) * (π)/2)));
   OCTANT2CONDITION = ((angle > (-0.25) * (π)) && (angle < 0));
   OCTANT3CONDITION = ((angle > 0) && (angle < (0.25 * (π))));
   OCTANT4CONDITION = ((angle > (0.25) * (π)) && (angle < ((0.50) * (π))));
   OCTANT5CONDITION = ((angle > (0.50) * (π)) && (angle < ((0.75) * (π))));
   OCTANT6CONDITION = ((angle > (0.75) * (π)) && (angle < (1 * (π))));
   OCTANT7CONDITION = ((angle > (-1) * (π)) && (angle < ((-0.75) * (π)/2)));
   OCTANT8CONDITION = ((angle > (-0.75) * (π)) && (angle < ((-0.50) * (π)/2)));
J. if(OCTANT1CONDITION)
         movex1 = 50;
            movey1 = movey1 + 15;
         line1_startx = line1_startx+movex1;
         line1_starty = line1_starty-movey1;


   end if
   if(OCTANT2CONDITION)
         movex2 = 50;
         movey2 = movey2 + 15;
            line1_startx = line1_startx+movex2;
            line1_starty = line1_starty+movey2;
```

```
end if
if(OCTANT3CONDITION)
      movex3 = 100;
      movey3 = movey3 + 15;
      line1_startx = line1_startx+movex3;
      line1_starty = line1_starty-movey3;
end if
if(OCTANT4CONDITION)
    movex2 = 25;
      movey4 = movey4 + 13;
      line1_startx = line1_startx-movex4;
      line1_starty = line1_starty+movey4;
end if
if(OCTANT5CONDITION)
    movex2 = -50;
      movey5 = movey5 + 13;
      line1_startx = line1_startx+movex5;
      line1_starty = line1_starty+movey5;
end if
if(OCTANT6CONDITION)
    movex2 = -100;
      movey6 = movey6 + 13;
```

```
          line1_startx = line1_startx-movex6;
          line1_starty = line1_starty-movey6;
   end if
   if(OCTANT7CONDITION)
          movex7 = -50;
          movey7 = movey2 + 15;
          line1_startx = line1_startx-movex2;
          line1_starty = line1_starty+movey2;
   end if
   if(OCTANT8CONDITION)
          movex8 = 0;
          movey8 = movey2 - 15;
          line1_startx = line1_startx+movex1;
          line1_starty = line1_starty-movey1;
   end if

K. context.moveTo(line1_startx,line1_starty);
   context.lineTo(line1_startx,line1_starty);
L. context.fillText(comp) – Label the enzyme at the end of the drawn line
M. End procedure
```

The main reason for dividing the circular map into octants and not as quadrants is to avoid the situation of overlapping labels in a quadrant where a large number of enzymes are grouped together. The reason for using positive and negative shifts in coordinates is to reverse the direction of labeling every octant. This allows enough lateral spacing to the labels that hover around the borders of each octant. The values of the displacement constants depend on the size of the circular map drawn onto the canvas and is computed dynamically for every execution. An example output from a test run is shown below in Figure 4.3.

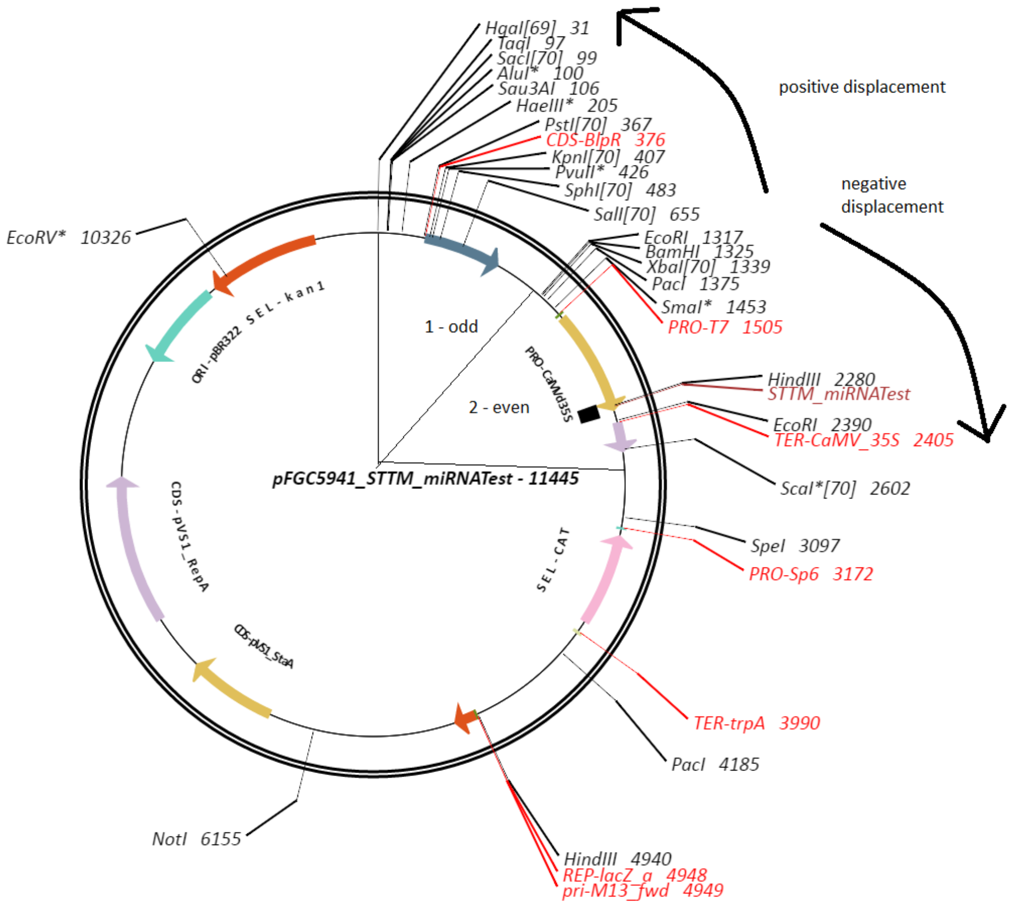


Figure 4.3: A sample plasmid map generated by Blossom STTM Hub with various labeling.

5 Automatic Annotation System using Exact Search and Basic Local Alignment Search Tool (BLAST)

After designing the STTM sequence, it must be incorporated into the destination vector sequence and the resulting sequence must be plotted into a circular plasmid. To accomplish this, two important functions need to be realized:

- To locate the one or two selected restriction enzymes in the initial vector binary sequence at which the designed STTM sequence should be inserted into. The designed STTM can gain the same restriction enzyme site or two different restriction enzyme sites at two ends by PCR amplification with the restriction enzyme sites to be incorporated into the primers. Both the amplified STTM sequence and the destination vector sequence were cut by one or two selected enzymes, and linked into a circular DNA. This procedure was achieved by mirna.pl PERL script.
- To automatically localize and annotate all the genes and restriction enzyme sites located in the destination vector sequence. This is achieved by blast.pl PERL script and stripos() PHP function.

5.1 Targeting the Designed STTM Sequence into the Destination Vector – mirna.pl

The mirna.pl PERL script is called for execution from Index3Copy2 PHP script using the PHP command exec(). The syntax to execute mirna.pl script is as follows:

exec("perl mirna.pl <restrictionenzyme1> <restrictionenzyme2> <STTM sequence> <initial vector> ","<OUTPUT>");

where,

a. Restrictionenzyme1 – First restriction enzyme

b. Restrictionenzyme2 – Second restriction enzyme (optional parameter)

c. STTM sequence – The designed STTM sequence

d. Initial vector – Initial vector sequence selected by a user

e. OUTPUT – The returned destination vector after the PERL script execution is completed

The mirna.pl PERL script searches for the location of the restriction enzyme site(s) in the initial vector sequence. If only one restriction enzyme is selected, the STTM sequence amplified from PCR with same restriction enzyme sites at two ends was placed at the location of this restriction enzyme site, and then returned back to the exec() call in the OUTPUT variable. If two restriction enzyme sites are selected, both are located in the initial vector sequence and the shorter sequence in between these two selected restriction enzyme sites was removed and replaced with the designed STTM sequence amplified from PCR with two different restriction enzyme sites at two ends of the STTM sequence. The PERL script returns the destination vector sequence containing the inserted STTM sequence back to the exec() call in the OUTPUT variable.

5.2 Annotating the Restriction Enzyme Sites Present in Destination Vector Sequence

Most of the restriction enzymes are small sequences and can be searched for exact matches in the destination vector. NEB provides a list of all the known restriction enzymes sites called

REBASE [14]. Using this list of restriction enzymes from REBASE, a table named ‘enzyme_list’ was formed with three columns – the first one(‘enzyme_name’) contains the name of the enzyme, the second one(‘start_seq’) contains the 5’ sequence of the enzyme and the third column(‘end_seq’) contains the 3’ sequence of the enzyme. The destination vector is searched for the presence of these restriction enzymes and if an exact match is found, the name and position of the enzyme in the destination vector are added to two arrays named enzyme[] and pos[]. These two arrays will be used to plot the enzymes in the plasmid map and then to label them. To search for the presence of restriction enzymes in the destination vector, the PHP function ‘stripos()’ - that uses linear search algorithm - was used.

Syntax of stripos(): stripos(searchstring, findstring, startposition)

Where, ‘searchstring’ is the string to search for the substring, ‘findstring’ is the string to look for in the search string and ‘startposition’ is the position in the searchstring to start finding a match for the findstring.

The following steps are involved in annotating the restriction enzymes in the destination vector:

- Retrieve all the enzymes stored in the ‘enzyme_list’ database using an SQL query

```
-----------------------------------------------------------------------------------------------
sql_query = “select * from enzyme_list”;
sql_statement = mysqli_prepare(db_connection, sql_query);
mysqli_stmt_execute(sql_statement);
mysqli_stmt_store_result(sql_statement);
mysqli_stmt_bind_result (sql_statement, enzyme_name, enzyme_sequence);
-----------------------------------------------------------------------------------------------
```

where db_connection is the reference to the connected database, mysqli_prepare() and mysqli_stmt_execute() functions execute the SQL query from PHP. sql_statement is the reference object where the results returned by mysqli_stmt_store_result() function from the SQL query are stored and the mysqli_stmt_bind_result() splits each column of the results in separate arrays (in this case the enzyme name and enzyme sequence).

```
• While(mysqli_stmt_fetch(sql_statement))
{
    returned_position = stripos(destination_vector_sequence,enzyme_sequence,0);
    if (returned_position != FALSE)
    {
        Store position of enzyme in destination vector to pos[] array
        Store name of enzyme sequence to enzyme[] array
    }
}
```

The above Pseudo code uses the stripos() method and completes the automatic annotation process of the enzymes present in the destination vector by storing the enzyme names and position in the destination vector to enzyme[] and pos[] arrays respectively.

5.3 Annotating the Larger Features (Genes, Promoters and Terminators) in Destination Vector Sequence Using blast.pl

To annotate the larger fragments that contain genes and other features present in the destination vector, it was not efficient to use a direct linear search as these fragments are very long and hence slow the process of finding exact matches in the destination vector sequence. In order to annotate these larger gene fragments, promoters, terminators and other features, we make use of the percentage matching algorithm BLAST *search.* After the destination vector sequence is returned from mirna.pl PERL script, it is passed to another PERL script named blast.pl that identifies all the genes and other features present in the sequence. PHP command shell_exec() is used to execute blast.pl. Before calling the blast.pl program, the destination vector sequence is stored in a temporary file called "qry.txt". Syntax for calling blast.pl: shell_exec("perl blast.pl"); Blast.pl reads the input vector sequence from "qry.txt" and performs a BLAST search [15] with the query sequence to determine the matching fragments present in the vector. A BLAST search reads in the query, parameters for the search and the database for identifying matches. The database that has been used in the BLAST search is a list of all the popularly known gene fragments obtained from the NCBI's reference sequences [16], a non-redundant sequence database of genomes, transcripts and proteins "feature.fasta.nt". This list has about 346 sequences of gene fragments (consisting of genes, promoters, terminators and other features). Each gene fragment from this list is in fasta format. The "vecdb" database is created by extracting each gene fragment from the "feature.fasta.nt" file. The fragment name and the fragment sequence form the two columns in "feature_list" table of the "vecdb" database. BLAST search offers the Command line BLAST search query blastn that is used in this application. The

BLAST search first checks the query for low complexity repeats and divides the large query into smaller words to be checked in detail. As this the query sequence is divided into smaller words, the matches from the databases are also initiated. In this preliminary search, the database is first scanned for matches in the divided words of the query to a subject sequence in the database where all the currently available vector sequence components are stored. When multiple words are found match a subject sequence in the database, the BLAST search extends to the two sides of each word matches. If exact match is not found, the percentage of gap free extensions is computed for every field in the database and is assigned a score. The lower scoring matches are deleted if too many matches are found. The highest scoring matches are then returned as the possible matches found in the overall query. The matches found in the destination vector are written to a file named ‘match_su.txt’.

The syntax used for using the command line BLAST search ‘blastn’ is as follows:

```
`blastn -query qry.txt -db vecdb -out match_su.txt -outfmt "<DETAILS OF THE EACH ROW
OF MATCH FOUND>" -task "blastn-short"`;
```

Where,

a. qry.txt – input of the destination vector sequence

b. vecdb – The database of the list of the fragments to be searched by BLAST extracted from the list of 346 fragments consisting of genes, promoters and terminators from the file ‘feature.fasta.nt’

c. match_su.txt – The output file where the matches found after BLAST search are stored.

d. <DETAILS OF EACH OF MATCH FOUND> - this is the list of genes or other features after they have been identified as matches by BLAST search. It includes fields like length of fragment, gene sequence, starting and ending position of the gene fragment in the destination vector, percentage of the gene or feature fragment sequence matching in the vector and the orientation of the gene or feature fragment. After the BLAST search is completed, the blast.pl reads the results returned in ‘match_su.txt’ file. For each gene fragment in this file, it’s sequence, it’s starting and ending positions in the vector and orientation are hashed to form a tabular structure and the table is written to a file called ‘feature_list.txt’. This ‘feature_list.txt’ file is returned as the final output of ‘blast.pl’ script. Index3Copy2 then reads, identifies and labels the gene fragments present in the destination vector from this file to plot the plasmid map.

5.4 Automatic Annotation System – Labeling Features in the Destination Binary Vector

As discussed in the previous section, the BLAST tool identifies the features contained in the destination plasmid vector and stores the resultant list in “feature.txt”. It is important to label the genes, promoters, terminators and enzymes of the destination vector on the plasmid map so that they can be identified by a user. The larger fragments are labeled inside the circular plasmid map along the arc whereas the smaller fragments and the restriction enzyme sites are labeled outside the circular plasmid map.

5.4.1 LABELING FEATURES INSIDE THE CIRCULAR PLASMID

It is not feasible to label all the features inside the circular plasmid map due to space limitations. Only features that are at least 5% as long as the destination vector, are labeled inside the plasmid along the arc. That is, feature will be named inside the plasmid if:

(end position of feature – start position of feature) >= (5/100)*length of plasmid

The list of features that satisfy this condition are selected from the BLAST results and this list is sent as an array parameter to the DrawTextAlongArc() function which also take starting position and ending position of the features as additional parameters. The algorithm to print the label along the arc between start and end positions of a feature is discussed in section 4.1.2 above. The implementation of this algorithm is shown in Figure 5.1.

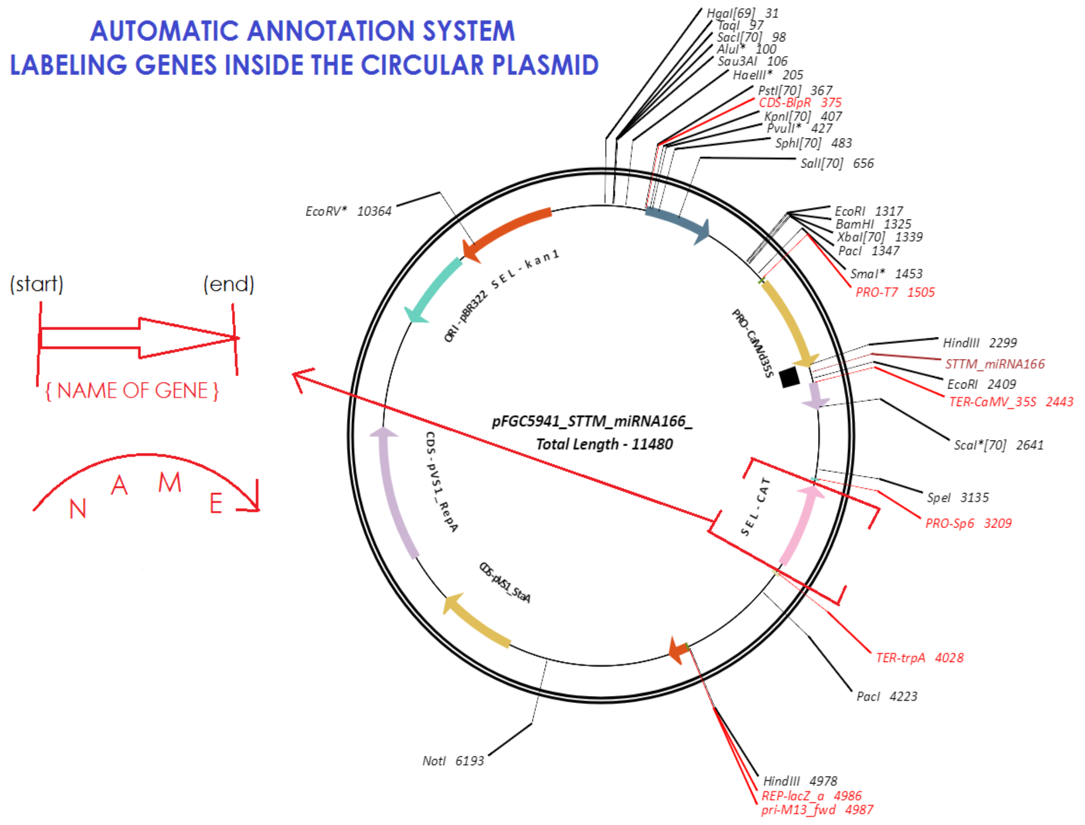


Figure 5.1: Automatic Annotation System – Labeling the features inside the circular plasmid along the arc. This uses the function DrawTextAlongArc() that computes the orientation and position of each character of the feature name.

5.4.2 Labeling Enzymes Outside the Plasmid Map

All the smaller fragments that are lesser than 5% of the total length of the destination vector and all the restriction enzymes are labeled outside the circular plasmid map. The challenging part in labeling these features is to ensure that no two labels overlap with each other. Since the destination vector could contain various enzymes depending on the initial vectors used, it is not possible to determine which will be the regions where more number of enzymes exists close to each other. To tackle this challenge we make use of the octant division labeling algorithm discussed in section 4.2.1. draw_enzyme() function is used to implement this algorithm. The following steps occur during the labeling of enzymes outside the circular plasmid map:

- The circular plasmid is divided into 8 equal octants of 45 degrees each.
- All the smaller features and restriction enzymes that have to be labeled are divided into 8 groups based on which octant they are present.
- Octants I, III, V and VII use positive(+ve) displacement method whereas octants II, IV, VI and VIII use negative(-ve) displacement method.
    - What is positive displacement method: The features in this octant are labeled in the anticlockwise direction. The angle in a canvas circle in HTML5 increases from 0 to 360 in anticlockwise direction and hence the name +ve displacement. The first feature has the shortest label line (line from the position of feature in the plasmid map to the feature's name in the label) and the least angle in the octant. The successive features have sequentially longer label lines which increase in constant displacements and are oriented at a higher angle than the previous feature.
    - What is negative displacement method: The features in this octant are labeled in the clockwise direction. In this method, the first feature has the shortest label line and the highest angular orientation to the circular plasmid. Successive features have sequentially longer label lines and lesser angle of orientation than the previous feature. The equations to these variables can be found in the draw_enzyme() function description in section 4.2.1.

- Once the label lines are drawn for all the features, the name of the feature is printed in a straight line without any angular orientation used to label the larger features.
- The octants use positive and negative displacement methods alternatively in order to avoid overlaps between the features that exist near either side of the octant's border.
- The implementation of this algorithm is shown in Figure 5.2.

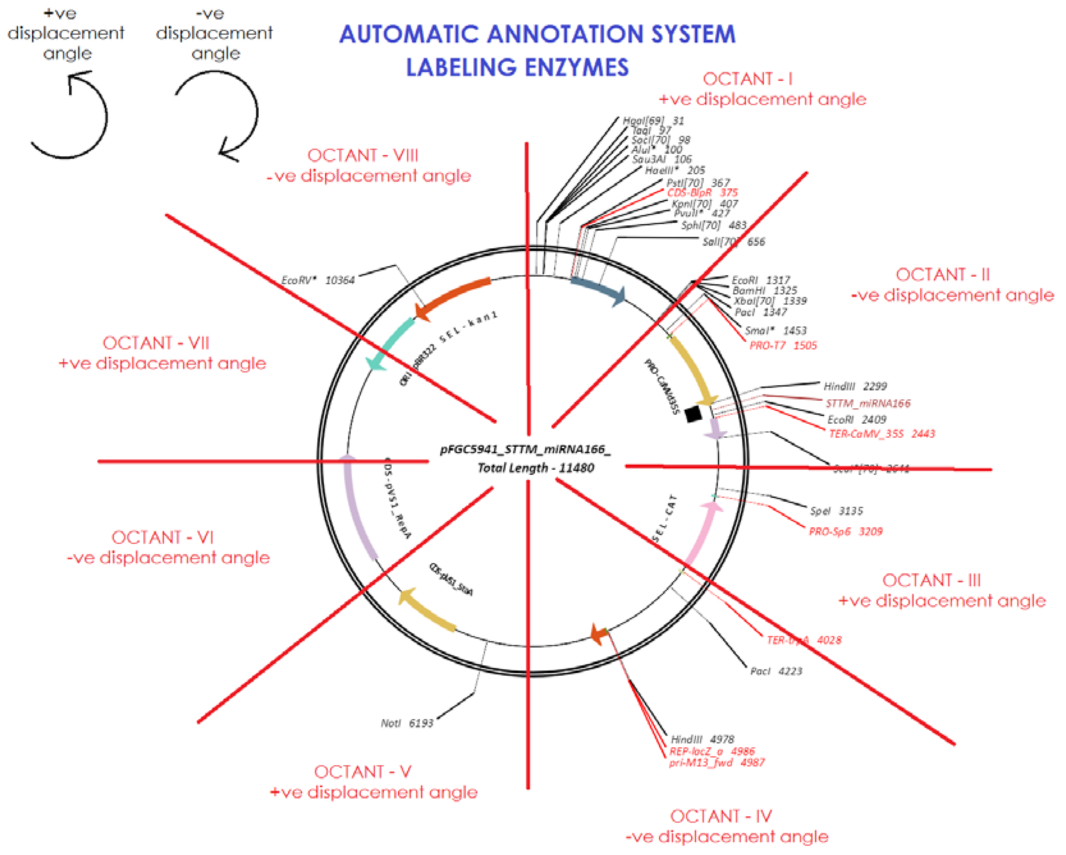


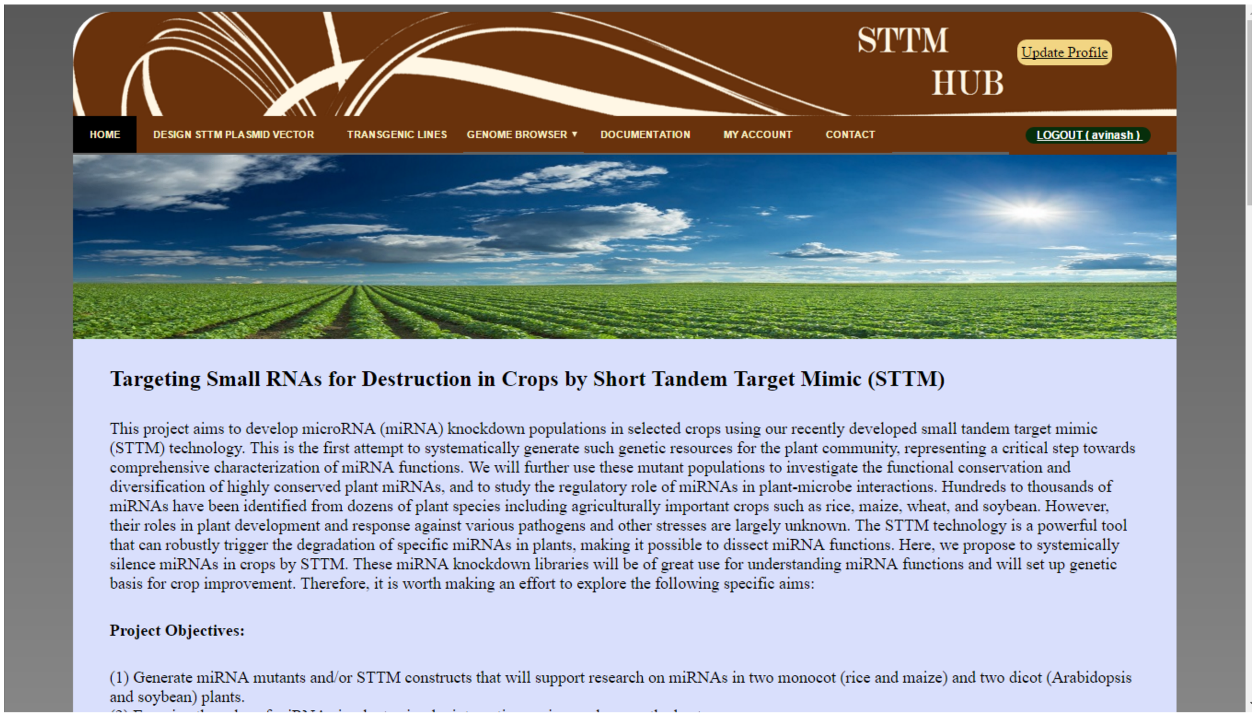
Figure 5.2: Automatic Annotation System – Labeling the features outside the circular plasmid map. This uses the function draw_enzyme() that divides the circular plasmid into equal octants and labels the features with positive and negative displacement methods. Octants that are set to use positive displacement method label the features in anticlockwise direction. Octants using negative displacement method label the features in clockwise direction.

6 Implementation

This section explains how all the Web pages in the Blossom STTM Hub were developed and the underlying work flow behind each web page. It also depicts the validation used in each of the inputs obtained from a user. The figures used in this section are all screenshots taken from the Blossom STTM Hub at corresponding stages of execution.

6.1 Homepage

The Blossom STTM Hub's homepage acts as a welcome page for users, and it contains essential information about the Short Tandem Target Mimic (STTM) project funded by National Science Foundation. The objectives of the project, the critical experimental approaches developed, the resources for looking up more regarding the project, and the participants involved are provided at the homepage. Figure 6.1 shows a screenshot of the Blossom STTM Hub's homepage.



Targeting Small RNAs for Destruction in Crops by Short Tandem Target Mimic (STTM)

This project aims to develop microRNA (miRNA) knockdown populations in selected crops using our recently developed small tandem target mimic (STTM) technology. This is the first attempt to systematically generate such genetic resources for the plant community, representing a critical step towards comprehensive characterization of miRNA functions. We will further use these mutant populations to investigate the functional conservation and diversification of highly conserved plant miRNAs, and to study the regulatory role of miRNAs in plant-microbe interactions. Hundreds to thousands of miRNAs have been identified from dozens of plant species including agriculturally important crops such as rice, maize, wheat, and soybean. However, their roles in plant development and response against various pathogens and other stresses are largely unknown. The STTM technology is a powerful tool that can robustly trigger the degradation of specific miRNAs in plants, making it possible to dissect miRNA functions. Here, we propose to systemically silence miRNAs in crops by STTM. These miRNA knockdown libraries will be of great use for understanding miRNA functions and will set up genetic basis for crop improvement. Therefore, it is worth making an effort to explore the following specific aims:

Project Objectives:

(1) Generate miRNA mutants and/or STTM constructs that will support research on miRNAs in two monocot (rice and maize) and two dicot (Arabidopsis and soybean) plants.

Figure 6.1: Webpage displaying the main objectives of the DesignSTTM and MaterialSTTM Web portals.

6.2 Login, User Registration and Update Profile Information

This section explains the login functionality and the process of new user registration to the STTM Web portal.

6.2.1 Login

The Login page has two functional flows—one for new users to register to Blossom STTM Web portal and the other for registered users to login to the portal. Registered users use their username and password to login to the portal. The system identifies new users by unique usernames to avoid username redundancy. Figure 6.2 shows a screenshot of the login page.

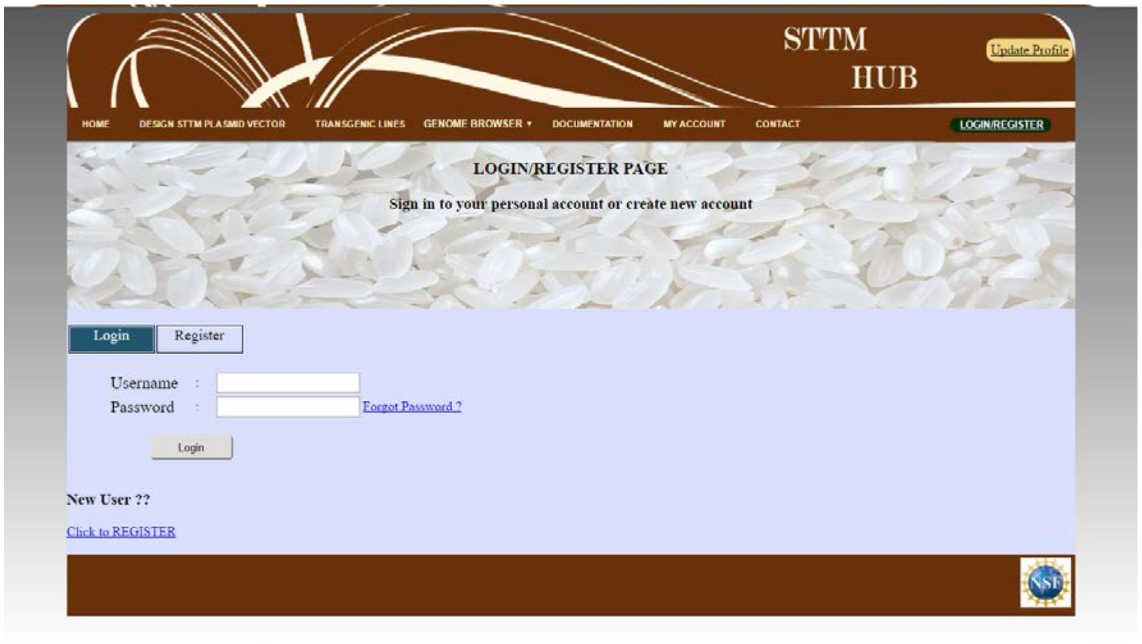
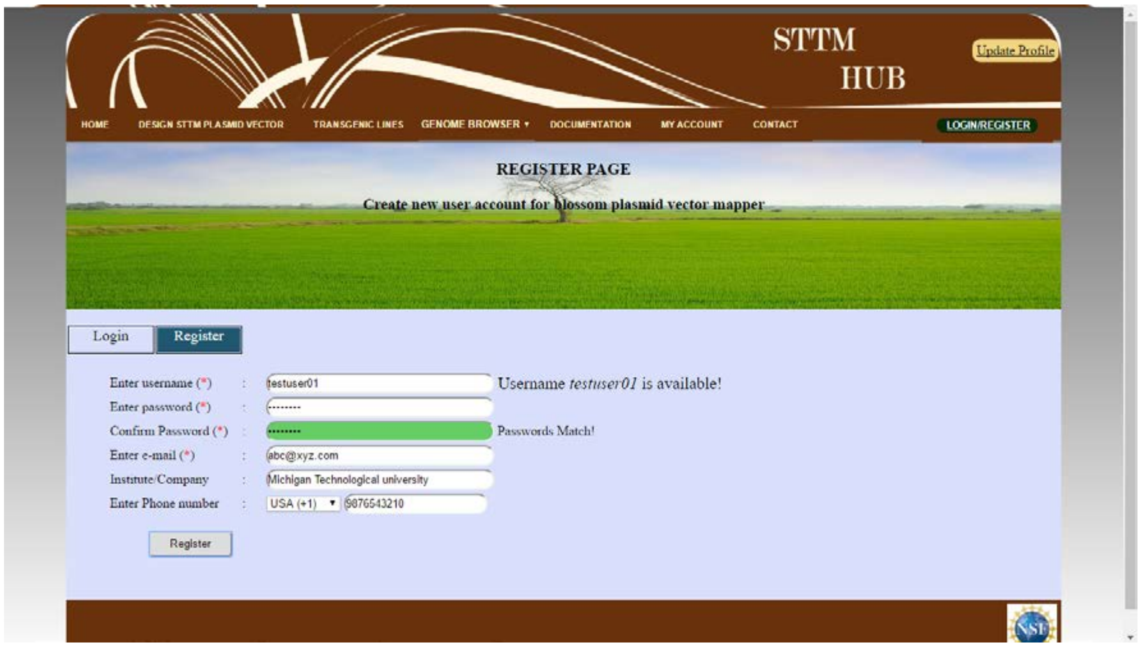


Figure 6.2: Blossom STTM Hub login page.

6.2.2 User Registration

The user registration process of Blossom STTM Hub was developed to obtain several user details (name, password, university, E-mail and Phone), validate them, and instantiate a new entry to the userdetails database containing the parsed input. JQuery plugins [25] were used to implement dynamic functions like checking for the availability of a username and correctness of the entered password in the "Confirm Password" text field. A user who intends to register to the Blossom STTM Hub is asked to provide the following information: username, password, and electronic mail address, institute or organization, and phone number. The username typed in by the user will be checked in the database, and the user will not be able to choose a username that already belongs to an existing user. The confirmation password field turns green only if the value entered matches with the password field's string. If it does not match, it stays in red for easy recognition. The username, password, and email fields are mandatory fields, whereas other fields are optional. The user can update his profile information after registration using the Update Profile feature of the portal. Figure 6.3 shows a screenshot that depicts the user registration process in Blossom STTM Hub.



REGISTER PAGE

Create new user account for blossom plasmid vector mapper

Enter username (*)	:	testuser01	Username *testuser01* is available!
Enter password (*)	:	••••••••	
Confirm Password (*)	:	••••••••	Passwords Match!
Enter e-mail (*)	:	abc@xyz.com	
Institute/Company	:	Michigan Technological university	
Enter Phone number	:	USA (+1) ▾ 9876543210	

Figure 6.3: Screenshot of web page illustrating new user registration process to Blossom STTM Web Portal.

6.3 Web Portal of DesignSTTM

The DesignSTTM Web Portal's main functions are to receive the various inputs such as the miRNA sequence, an intermediate vector, a destination vector, and restriction enzyme sites, and design the STTM sequence using these inputs and finally plot the destination plasmid DNA map. The DesignSTTM Web Portal comprises of two significant parts—designindex.php and index3copy2.php. Designindex.php script handles reception of inputs, whereas index3copy2.php script accomplishes the tasks such as utilizing the inputs to design STTM sequence structure and plotting the destination plasmid DNA containing the designed STTM sequence.

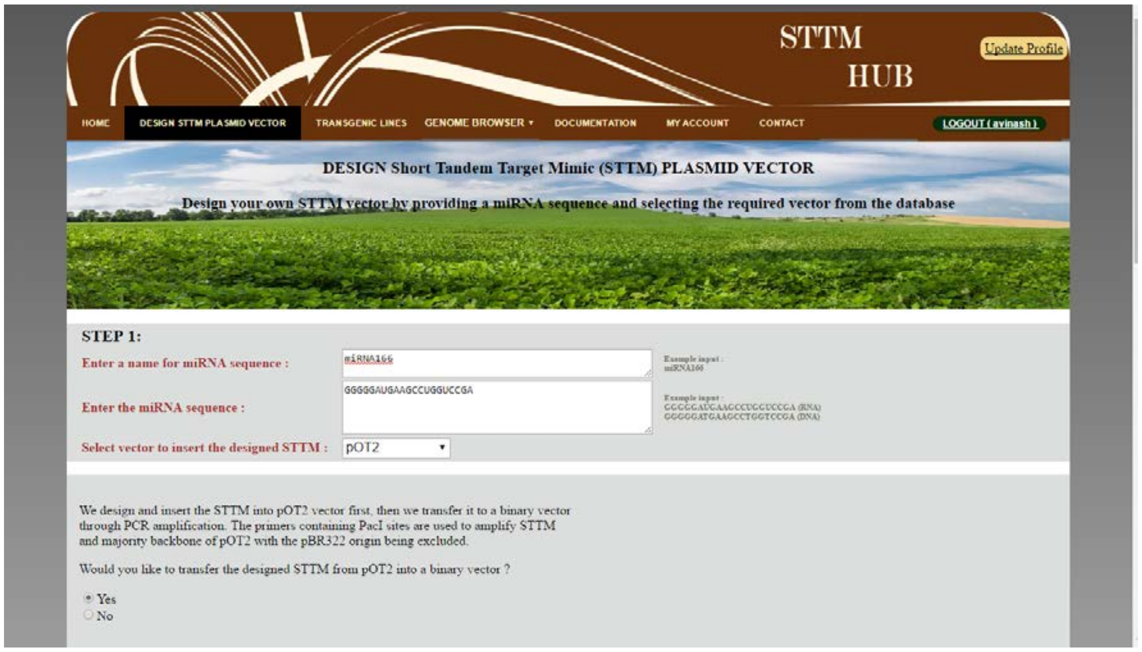
Designindex PHP script: Acquisition of inputs from a user to design STTM sequence

As mentioned above, the main functions of the "designindex" PHP script is to obtain miRNA sequence, intermediate vector, destination binary vector and the restriction enzyme sites from a user, validate these inputs, and pass them to the "index3copy2" PHP script to design the STTM sequence and plot the destination plasmid DNA map containing the STTM structure. The Designindex PHP script accomplishes the design in two steps. The first step is to obtain the miRNA sequence and the intermediate vector from the user. The second step is to obtain the destination binary vector chosen by the user in the circumstance in which the designed STTM structure needs transferring to a destination binary vector. The insertion of STTM structure into the destination binary vector occurs at one restriction enzyme digestion site, or between two restriction enzyme sites. The Designindex PHP script thereby needs to obtain one or two restriction digestion sites in the destination binary vector.

6.3.1 Step 1: Obtain miRNA Input and Select Intermediate Vector

- A user has to enter a name for the miRNA sequence for designing the STTM sequence. This name will be used as a reference in naming the STTM construct once it is inserted into a plasmid vector. The miRNA name will also be used for the filename in the output report, so that the user can be easily identified it for future reference.
    - Validation: The miRNA name field cannot be empty, and the system will stop the user from submitting the vector without a name.

- A user then enters the miRNA sequence of 18 to 30 characters long to be used to construct the STTM sequence. A miRNA sequence in DNA format (ATGC) or in RNA format (AUGC) can be entered, but the latter will be converted to DNA format automatically before it is to be used for designing STTM vector in DNA format.
    - Validation: The miRNA sequence entered by the user must contain only valid sequences with A, T, G, and C nucleotides or A, U, G, and C nucleobases(A-adenine, T-thymine, G-Guanine, and C-cytosine), and their lengths must vary between 18 and 30. If the user either provides an empty or invalid sequence, an error message is prompted, and the program will halt.



DESIGN Short Tandem Target Mimic (STTM) PLASMID VECTOR

Design your own STTM vector by providing a miRNA sequence and selecting the required vector from the database

STEP 1:

Enter a name for miRNA sequence : miRNA166

Example input : miRNA166

Enter the miRNA sequence : GGGGAUGAAGCCUGGUCCGA

Example input : GGGGAUGAAGCCUGGUCCGA (RNA) GGGGATGAAGCCTGGTCCGA (DNA)

Select vector to insert the designed STTM : pOT2

We design and insert the STTM into pOT2 vector first, then we transfer it to a binary vector through PCR amplification. The primers containing PacI sites are used to amplify STTM and majority backbone of pOT2 with the pBR322 origin being excluded.

Would you like to transfer the designed STTM from pOT2 into a binary vector ?

[✓ Yes] No

Figure 6.4: The web interface for submitting a miRNA name and sequence for designing STTM structure in DesignSTTM Web Portal.

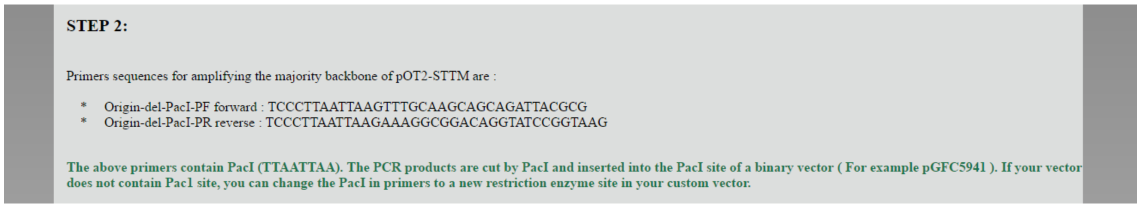
- Selection of a vector from the database: The DesignSTTM web interface enables a user to select an intermediate vector from a dropdown menu to insert the designed

STTM construct temporarily before being transferred to a destination binary vector. Based on current development, pOT2 vector, a custom intermediate vector, is used to harbor the STTM structure before it is transferred to the destination vector. In the future, more intermediate vectors may be added to the database and the dropdown list. Figure 6.4 shows screenshots of Step 1 of workflow for DesignSTTM.

- Once a user has provided the miRNA sequence and selected the intermediate vector, he is provided an option to transfer the designed STTM construct to a destination binary DNA vector that can be used to deliver a STTM structure or the majority of intermediate vector (except origin sequence) into plant cells. The origin sequence involves the regions where the circular plasmid initiates the duplication of itself. By transferring a STTM structure to a binary vector, we meant to use the PCR to amplify the majority of STTM sequence + pOT2 vector, and then insert the PCR product into a destination binary vector that the user chooses. The origin segment sequence in the pOT2 vector needs to be removed before transferring into a destination binary vector. This is because a binary vector cannot have more than one origin segment [4]. This step will determine the next step dynamically. If option “NO” is chosen, the designed STTM construct will be inserted into the intermediate vector at one of the restriction enzyme digestion sites or between two restriction enzyme digestion sites selected by the user, and the resulting plasmid DNA vector containing the STTM structure will be plotted to the map. If option “YES” is chosen, STEP 2 is triggered dynamically, in which the user selects the destination binary vector and the restriction enzyme sites to insert the majority backbone of the intermediate vector containing STTM sequence into the destination binary vector.

6.3.2 Step 2: Select Destination Binary Vector and Restriction Enzyme Sites

- If a user chooses "YES" to transfer the designed STTM construct into a destination binary vector, the origin segment has to delete in the intermediate vector before transferring to the binary vector. To achieve this, we designed two primers to amplify the majority of backbone of STTM structure present in non-binary vector like pOT2 (Origin-deletion PacI forward primer —TCCCTTAATTAAGTTTGCAAGCA GCAGATTACGCG and reverse primer – TCCCTTAATTAAGAAAGGCGGACAG GTATCCGGTAAG). The replicate origin skipped in pOT2 is 616 bp long. Figure 6.5 shows the primers to be used to amplify the majority backbone of the constructed STTM vector into the binary vector.



STEP 2:

Primers sequences for amplifying the majority backbone of pOT2-STTM are :

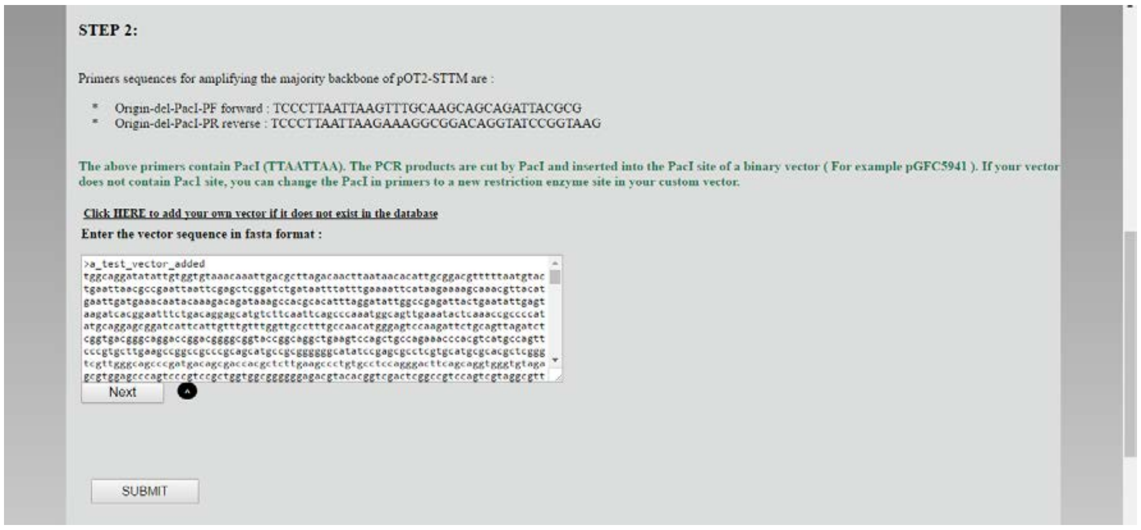
- * Origin-del-PacI-PF forward : TCCCTTAATTAAGTTTGCAAGCAGCAGATTACGCG
- * Origin-del-PacI-PR reverse : TCCCTTAATTAAGAAAGGCGGACAGGTATCCGGTAAG

The above primers contain PacI (TTAATTAA). The PCR products are cut by PacI and inserted into the PacI site of a binary vector (For example pGFC5941). If your vector does not contain Pac1 site, you can change the PacI in primers to a new restriction enzyme site in your custom vector.

Figure 6.5: Primers to amplify majority backbone of pOT2-STTM sequence for cloning it into a destination binary vector.

- Selecting the binary vector:
    - o A user can either choose from a list of binary vectors using the drop down menu or enter his own binary vector sequence, which would be added to his personal directory.

- The following actions take place in the background once the user chooses to enter his own vector sequence to the tool:
    - The sequence is checked to contain only adenine(A), thymine(T), Guanine(G), and cytosine(C) nucleobases, and it is saved to a file in the user's personal directory and the vector is added to the user's personal database.
    - Additionally, the sequence is annotated by running the BLAST [17] Perl script, and the enzymes and other features of the vector are also stored in the database. The user can now use the destination binary vector added to be targeted by the designed STTM sequence.
    - The user then selects the binary vector from the drop down menu. Figure 6.6 shows the process of adding a binary vector to transfer the constructed STTM sequence into it.



STEP 2:

Primers sequences for amplifying the majority backbone of pOT2-STTM are :

* Origin-del-PacI-PF forward : TCCCTTAATTAAGTTTGCAAGCAGCAGATTACGCG
* Origin-del-PacI-PR reverse : TCCCTTAATTAAGAAAGGCGGACAGGTATCCGGTAAG

The above primers contain PacI (TTAATTAA). The PCR products are cut by PacI and inserted into the PacI site of a binary vector (For example pGFC5941). If your vector does not contain PacI site, you can change the PacI in primers to a new restriction enzyme site in your custom vector.

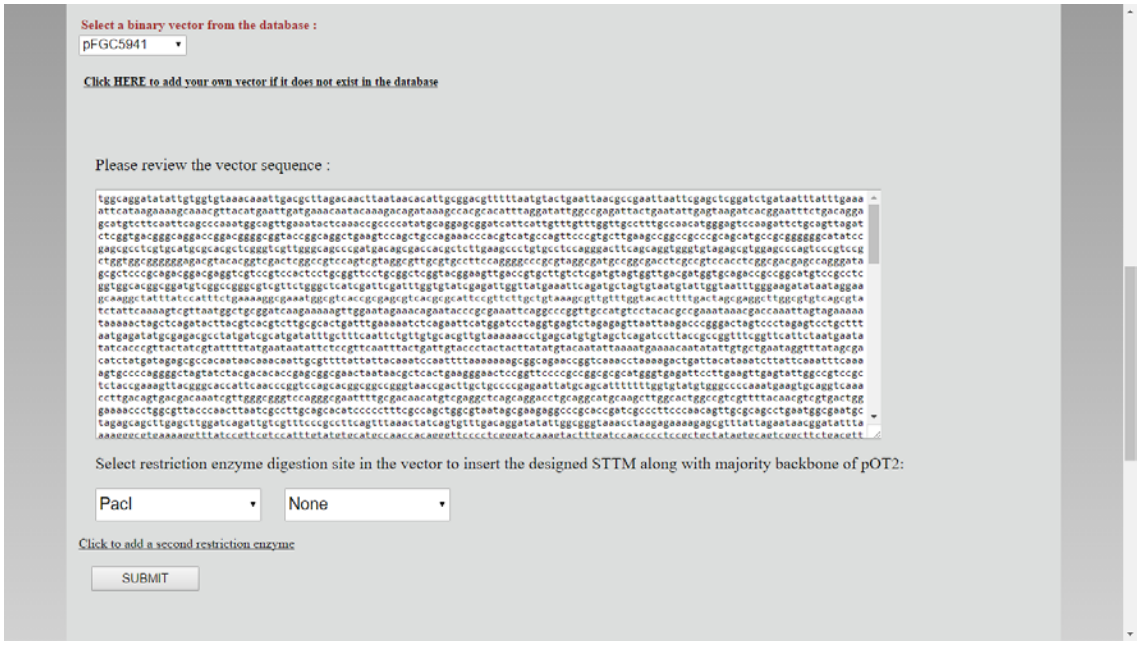
Click HERE to add your own vector if it does not exist in the database

Enter the vector sequence in fasta format :

```
>a_test_vector_added
tggcaggatatattgtggtgtaaacaaattgacgcttagacaacttaataacacattgcggacgtttttaatgtac
tgaattaacgccgaattaattcgagctcggatctgataatttatttgaaaattcataagaaaagcaaacgttacat
gaattgatgaaacaatacaaagacagataaagccacgcacatttaggatattggccgagattactgaatattgagt
aagatcacggaatttctgacaggagcatgtcttcaattcagcccaaatggcagttgaaatactcaaaccgccccat
atgcaggagcggatcattcattgtttgtttggttgcctttgccaacatgggagtccaagattctgcagttagatct
cggtgacgggcaggaccggacggggcggtaccggcaggctgaagtccagctgccagaaacccacgtcatgccagtt
cccgtgcttgaagccggccgcccgcagcatgccgcgggggcatatccgagcgcctcgtgcatgcgcacgctcggg
tcgttgggcagcccgatgacagcgaccacgctcttgaagccctgtgcctccagggacttcagcaggtgggtgtaga
gcgtggagcccagtcccgtccgctggtggcgggggagacgtacacggtcgactcggccgtccagtcgtaggcgtt
```

Figure 6.6: Addition of a binary vector to which the majority backbone of pOT2-STTM sequence can be transferred.

- When a user selects the binary vector from the drop down menu, the vector sequence is retrieved from database and displayed in a text area dynamically using jQuery inline script [18].
- The last step in designindex script is to select restriction enzyme digestion site(s) to insert the majority backbone of the intermediate vector that includes the constructed STTM construct into the destination binary vector. All the restriction enzymes contained in the binary vector are displayed in a drop down menu as a list for selection. A user can select either a single restriction enzyme site or two restriction enzyme sites. If one restriction enzyme site is selected, the STTM construct will be inserted at the location of this enzyme in the binary vector. If two restriction enzyme sites are selected, the STTM construct is inserted between these two enzymes in the destination binary vector. Figure 6.7 depicts the selection of restriction digestion enzyme site(s) in a binary vector, amplifying the designed STTM sequence in the intermediate vector, and then inserting the majority backbone of the intermediate vector into the destination binary vector.


Select a binary vector from the database :

pFGC5941

Click HERE to add your own vector if it does not exist in the database

Please review the vector sequence :

```
tggcaggatatattgtggtgtaaacaaattgacgcttagacaacttaataacacattgcggacgtttttaatgtactgaattaacgccgaattaattcgagctcggatctgataatttatttgaaa
attcataagaaaagcaaacgttacatgaattgatgaaacaatacaaagacagataaagccacgcacatttaggatattggccgagattactgaatattgagtaagatcacggaatttctgacagga
gcatgtcttcaattcagcccaaatggcagttgaaatactcaaaccgccccatatgcaggagcggatcattcattgtttgtttggttgcctttgccaacatgggagtccaagattctgcagttagat
ctcggtgacgggcaggaccggacggggcggtaccggcaggctgaagtccagctgccagaaacccacgtcatgccagttcccgtgcttgaagccggccgcccgcagcatgccgcggggggcatatcc
gagcgcctcgtgcatgcgcacgctcgggtcgttgggcagcccgatgacagcgaccacgctcttgaagccctgtgcctccagggacttcagcaggtgggtgtagagcgtggagcccagtcccgtccg
ctggtggcggggggagacgtacacggtcgactcggccgtccagtcgtaggcgttgcgtgccttccaggggcccgcgtaggcgatgccggcgacctcgccgtccacctcggcgacgagccagggata
gcgctcccgcagacggacgaggtcgtccgtccactcctgcggttcctgcggctcggtacggaagttgaccgtgcttgtctcgatgtagtggttgacgatggtgcagaccgccggcatgtccgcctc
ggtggcacggcggatgtcggccgggcgtcgttctgggctcatcgattcgatttggtgtatcgagattggttatgaaattcagatgctagtgtaatgtattggtaatttgggaagatataatggaa
gcaaggctatttatccatttctgaaaggcgaaatggcgtcaccgcgagcgtcacgcgcattccgttcttgctgtaaagcgttgtttggtacacttttgactagcgaggcttggcgtgtcagcgta
tctattcaaaagtcgttaatggctgcggatcaagaaaaagttggaatagaaacagaatacccgcgaaattcaggcccggttgccatgtcctacacgccgaaataaacgaccaaattagtagaaaaa
taaaaactagctcagatacttacgtcacgtcttgcgcactgatttgaaaaatctcagaattcatggatcctaggtgagtctagagagttaattaagacccgggactagtccctagagtcctgcttt
aatgagatatgcgagacgcctatgatcgcatgatatttgctttcaattctgttgtgcacgttgtaaaaaacctgagcatgtgtagctcagatccttaccgccggtttcggttcattctaatgaata
tatcacccgttactatcgtatttttatgaataatattctccgttcaatttactgattgtaccctactacttatatgtacaatattaaaatgaaaacaatatattgtgctgaataggtttatagcga
catctatgatagagcgccacaataacaaacaattgcgttttattattacaaatccaattttaaaaaaagcggcagaaccggtcaaacctaaaagactgattacataaatcttattcaaatttcaaa
agtgccccaggggctagtatctacgacacaccgagcggcgaactaataacgctcactgaagggaactccggttccccgccggcgcgcatgggtgagattccttgaagttgagtattggccgtccgc
tctaccgaaagttacgggcaccattcaacccggtccagcacggcggccgggtaaccgacttgctgccccgagaattatgcagcatttttttggtgtatgtgggccccaaatgaagtgcaggtcaaa
ccttgacagtgacgacaaatcgttgggcgggtccagggcgaattttgcgacaacatgtcgaggctcagcaggacctgcaggcatgcaagcttggcactggccgtcgttttacaacgtcgtgactgg
gaaaaccctggcgttacccaacttaatcgccttgcagcacatccccctttcgccagctggcgtaatagcgaagaggcccgcaccgatcgcccttcccaacagttgcgcagcctgaatggcgaatgc
tagagcagcttgagcttggatcagattgtcgtttcccgccttcagtttaaactatcagtgtttgacaggatatattggcgggtaaacctaagagaaaagagcgtttattagaataacggatattta
aaaggcgtgaaaaggtttatccgttcgtccatttgtatgtgcatgccaaccacagggttcccctcgggatcaaagtactttgatccaacccctccgctgctatagtgcagtcggcttctgacgtt
```

Select restriction enzyme digestion site in the vector to insert the designed STTM along with majority backbone of pOT2:

PacI

None

Click to add a second restriction enzyme

Figure 6.7: Depicts the selection of restriction digestion enzyme site(s) in binary vector and amplifying the designed STTM sequence in pOT2 with primers containing the enzyme site(s). The amplified fragment is then inserted into the binary vector.

- Upon reception of all the necessary inputs, the index3copy2.php script is called for execution.

6.4 Index3copy2 PHP Script: Design STTM Construct, Primers, and Plot Plasmid DNA map

The script of Index3copy2 was developed to realize the following functionalities: (1) designing the STTM construct; (2) designing forward and reverse primers to be used to amplify the STTM construct in the destination binary vector; (3) plotting the plasmid DNA vector map containing the designed STTM sequence, (4) displaying the final sequence and features, and send results as a PDF document to the user's email.

1) Map of destination vector map containing STTM structure
2) Sequence of destination vector
3) Features of destination vector
4) STTM Sequence
5) Electronic-mail REPORT

6.4.1 Map of destination vector map containing STTM structure

Generation of the plasmid map of the destination binary vector containing the designed STTM construct is the most critical step in index3copy2 PHP script. The inputs (miRNA sequence, intermediate vector, destination binary vector, and restriction enzyme sites) given by a user in designindex script are stored in PHP session variables and are passed on to the index3copy2 PHP script. The STTM sequence is designed based on these inputs. This is followed by the construction of forward and reverse primers that can be used to amplify the designed STTM sequence using Polymerase Chain Reaction(PCR) [19] and then transfer it into a destination binary vector. A binary vector is a circular DNA that can serves as a vehicle for delivering the gene to plant genome. Finally, the designed STTM construct is inserted into the destination binary vector at the restriction enzyme's location, and the plasmid DNA map is plotted using HTML Canvas.

- STTM sequence design:
    - The STTM sequence is designed by a Perl sub-routine mirna.pl that is executed from index3copy2 PHP script. The mirna Perl sub-routine takes in the two restriction enzymes, the miRNA input sequence and the intermediate vector sequence from the stored SESSION variables and incorporates the following algorithm to design the STTM sequence:

      Let us take the design of STTM165/166-48nt as an example:

The miR165 sequence is: 5’ UCGGACCAGGCUUCAUCCCCC 3’

and the miR166 sequence is 5’ UCGGACCAGGCUUCAUUCCCC 3’

→ Reverse compliment the miRNA input sequence

The anti-paralleled miR165 is: 5’ GGGGGAUGAAGCCUGGUCCGA 3’

The anti-paralleled miR166 is: 5’ GGGGAAUGAAGCCUGGUCCGA 3’

→ Then replace the NNN sequences with the reverse complimented miRNA sequences.

5’catttggagaggacagcccAAGCTTGGGGGAUGAAGCCUGGUCCGAgttgttgttgtta tggtctaatttaaatatggtctaaagaagaagaatGGGGAAUGAAGCCUGGUCCGAGAATT Cggtacgctgaaatcaccag 3’

→ Add a bulge sequence with a length of 3 nucleotides in between the miRNA sequence. The 3 nucleotide bulge sequence can be any random sequence except for the condition that it must not be the same as the compliment of the sequence between the 11th and 13th nucleotides of the miRNA sequence. The bulge sequence is placed so that the binding sites could trap miRNAs without being cleaved. The bulge sequence is added between the 10th and 11th nucleotide of the miRNA sequence.

5’catttggagaggacagcccAAGCTTGGGGGAUGAAGctaCCUGGUCCGAgttgttgttg ttatggtctaatttaaatatggtctaaagaagaagaatGGGGAAUGAAGctaCCUGGUCCGAGA ATTCggtacgctgaaatcaccag 3’ (In this case the chosen bulge sequence is ‘cta’)

→ Split the spacer sequence into two halves using the SwaI enzyme location as the midpoint. The first half of the spacer is to be added into the forward primer containing the first half of the STTM sequence while the second half of the spacer is added into the reverse primer containing the second-half of the STTM sequence.

→ Finally the STTM sequence is constructed by concatenating the following calculated parameters:

STTM sequence = first restriction enzyme +

Reverse complimented miRNA sequence +

Spacer sequence + reverse complementary miRNA(same or different miRNA)

Sequence + second restriction enzyme +

Spacer sequence

In the above STTM design method, either one or two different miRNA sequences can be targeted to form the STTM vector. If only one miRNA is used for designing STTM structure, then the STTM sequence contains two copies of complimented sequences of the same miRNA at both two loci that are separated by spacer sequence. If two different miRNA’s are targeted, normal sequence of the first miRNA and anti-paralleled sequence of the second miRNA are used to design the STTM sequence.

→ The designed STTM sequence is inserted into the intermediate vector at the location between two the restriction enzyme sites, as shown in Figure 6.8. The resultant intermediate vector sequence containing the designed STTM sequence is stored in a text file in the user's personal directory.

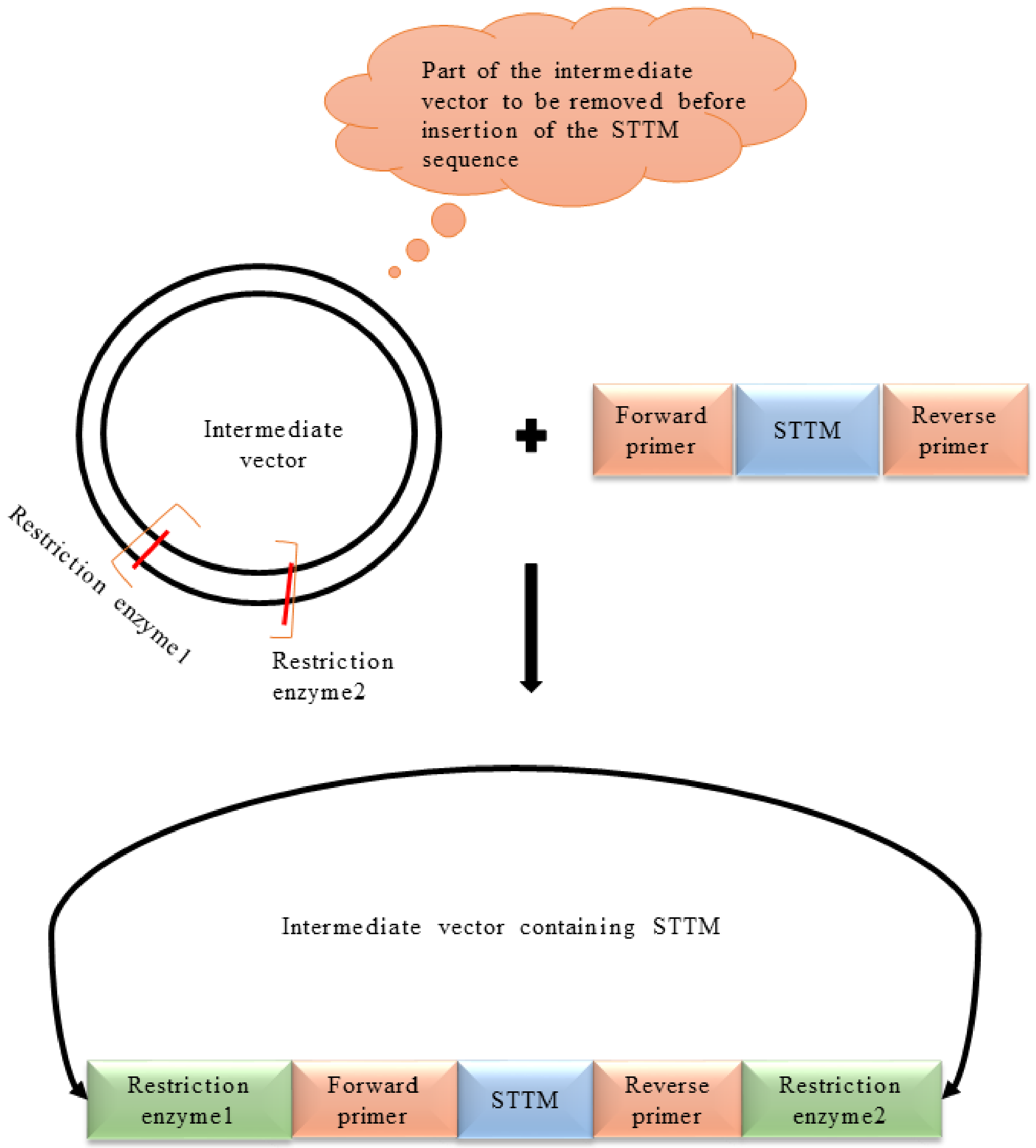


Figure 6.8: STTM construct design and insertion of designed STTM sequence into the destination vector.

- Designing forward and reverse primers to amplify the STTM structure:

  - Forward and reverse primers were designed to amplify the designed STTM construct into the destination binary vector through PCR (Polymerase Chain Reaction) [19]. PCR is a technique that is widely used in molecular biology when there is a need to amplify a DNA fragment into multiple copies of several orders of magnitude. In our case, we use PCR to amplify the designed STTM sequence and the majority of intermediate vector (except replication origin) into a destination vector sequence. The DesignSTTM Web Portal computes the forward and reverse primers, and the sequences are also displayed to users. The detailed procedure for designing the forward and reverse primer sequences are listed below.

- Forward primer:
  - Copy the second half of the designed STTM sequence.
  - Add a few bases GCC to the 5' end of primer so that the restriction enzyme site at the 5'end is protected.
  - Replace any U's in the sequence with T's.
- Reverse primer:
  - Copy the first half of the designed STTM sequence
  - Reverse the sequence
  - Complement the reversed sequence - change A->T, T->A, G->C, C->G
  - Add the 5'end protection sequence to the 5's end of reverse primer sequence so that the restriction enzyme sites at the 5'end to be protected.
  - Replace any U's in the sequence with T's

- An example of designing STTM structure :
    - **Designed STTM sequence with miRNA166:**

      catttggagaggacagccc**AAGCTT**TCGGACCAGGCgtaTTCATCCCCCgttgttgttgttat ggtctaatttaaatatggtctaaagaagaaggatTCGGACCAGGCgtaTTCATCCCCC**GAATT C**ggtacgctgaaatcaccag

      Forward primer:

      GCCatttaaatatggtctaaagaagaaggatTCGGACCAGGCgtaTTCATCCCCC**GAATT C**ggtacgctgaaatcaccag

      Reverse primer:

      GCCatttaaattagaccataacaacaacaacGGGGGATGAAtacGCCTGGTCCGA**AAGCT T**gggctgtcctctccaaatg

      Restriction Enzymes used in the above example: HindIII – AAGCTT & ECoRI - GAATTC

- Designed STTM construct has to be transferred into a destination binary vector from an intermediate vector, and then the replication origin segment presents in the intermediate vector needs to be deleted because the destination binary vector has its own replication origin.

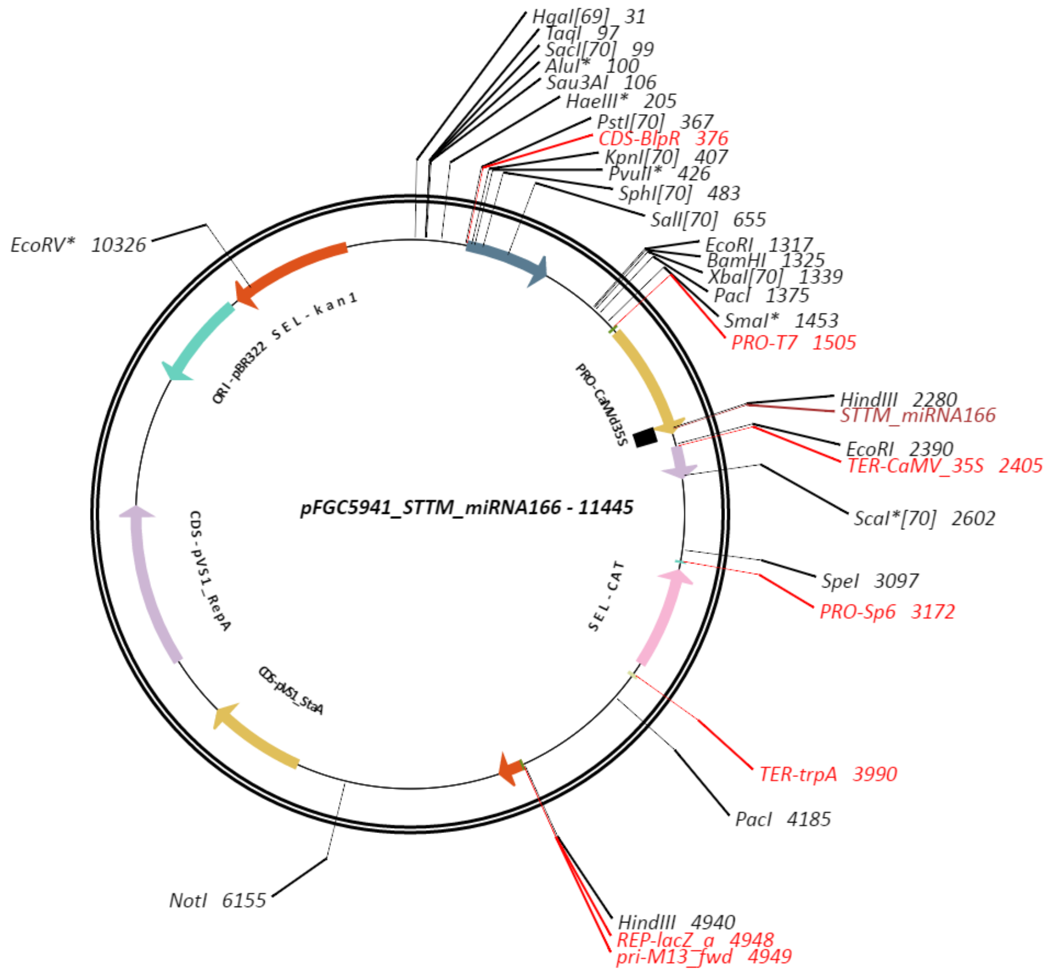


Figure 6.9: Plasmid DNA map of the binary vector pFGC5941 containing the designed STTM structure depicting all the enzymes, genes, and features present in the destination binary vector

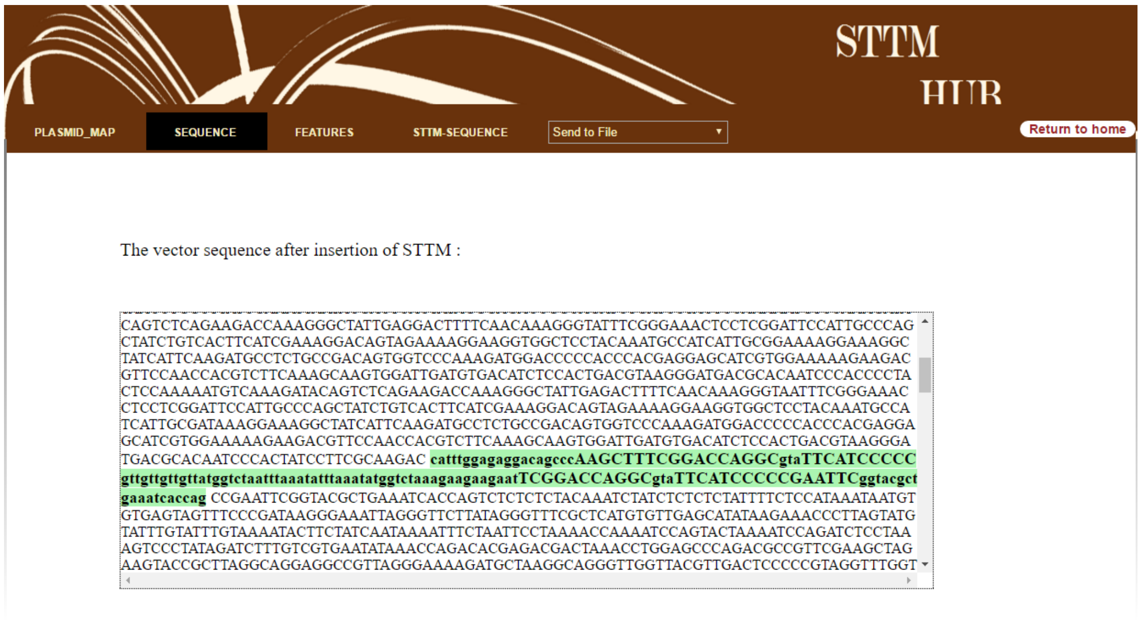
- If a user selects one restriction enzyme site to insert the designed STTM construct, the system searches the site in the destination binary vector. Once this location containing the site is determined, the intermediate vector sequence (without replication origin) containing the designed STTM sequence is retrieved and inserted into the site.

- If a user selects two restriction enzyme sites, the intermediate vector with the designed STTM sequence is inserted between the restriction enzymes.
- Again, blast.pl is executed to annotate the genes, promoters and terminators, and stripos() PHP method is used to annotate the restriction enzyme sites contained in the resulting destination binary vector after the insertion of the STTM construct. The results are written to the text file “feature_list.txt”.
- The enzyme names, positions in the sequence, and their orientations are stored in distinct arrays(enzyme[] and pos[]) that are used later to plot the plasmid map onto the canvas [20].
- Now that all the data required for the plasmid map plotting have been prepared, index3copy2 script plots the plasmid map using HTML canvas element, labels the enzyme sites, and shows their orientations.
- DRAW_ARC() is the function that maps the features of the STTM containing plasmid vector onto the canvas.
- DrawTextAlongArc() is the function which is used to label the features along the circular ring. Only features that are longer than one-twentieth of the overall vector length are labeled inside the circular ring in the map. This is done to avoid overriding of labels when trying to plot smaller features that are closer to each other in the vector.

- DRAW_ENZYME_ARC() plots and labels the enzymes contained in the destination binary vector.
- Figure 6.9 shows a sample plasmid map plotted by the DesignSTTM Web Portal.

6.4.2 Sequence

"Index3Copy2" PHP script displays the destination binary vector sequence obtained after inserting the STTM construct from the intermediate vector to the binary vector. The STTM part of the destination vector sequence will be highlighted to facilitate easy identification. Figure 6.10 shows the STTM vector sequence as displayed in the results page.



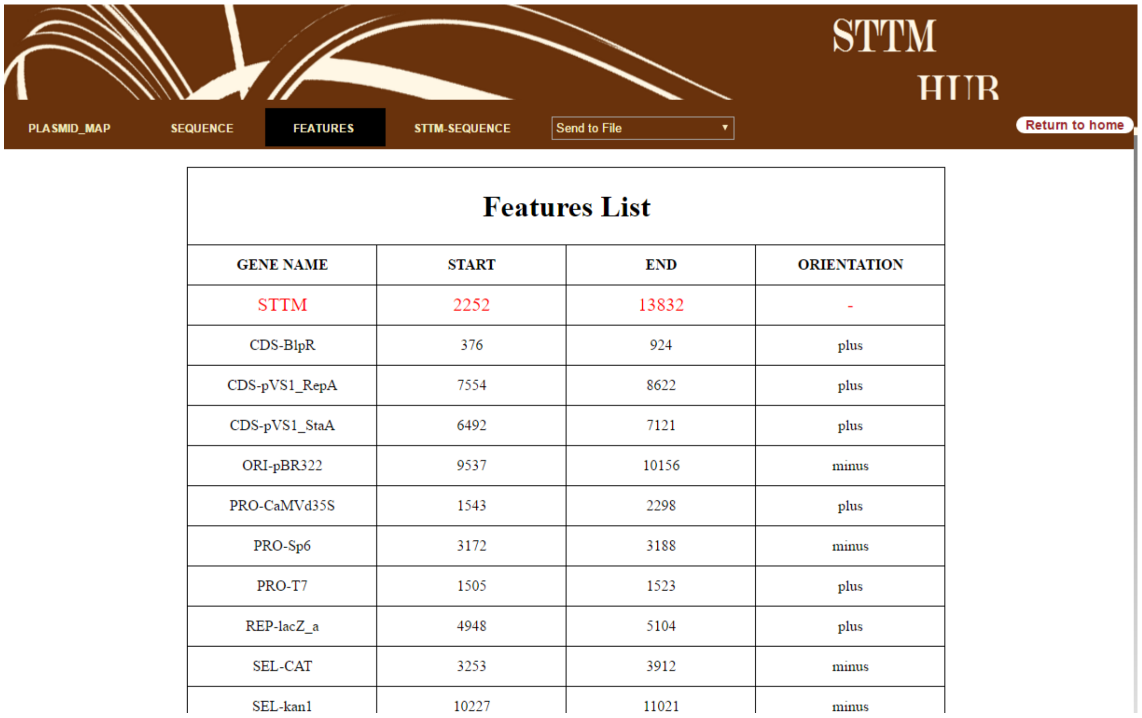
The vector sequence after insertion of STTM :

```
CAGTCTCAGAAGACCAAAGGGCTATTGAGGACTTTTCAACAAAGGGTATTTCGGGAAACTCCTCGGATTCCATTGCCCAG
CTATCTGTCACTTCATCGAAAGGACAGTAGAAAAGGAAGGTGGCTCCTACAAATGCCATCATTGCGGAAAAGGAAAGGC
TATCATTCAAGATGCCTCTGCCGACAGTGGTCCCAAAGATGGACCCCCACCCACGAGGAGCATCGTGGAAAAAGAAGAC
GTTCCAACCACGTCTTCAAAGCAAGTGGATTGATGTGACATCTCCACTGACGTAAGGGATGACGCACAATCCCACCCCTA
CTCCAAAAATGTCAAAGATACAGTCTCAGAAGACCAAAGGGCTATTGAGACTTTTCAACAAAGGGTAATTTCGGGAAAC
CTCCTCGGATTCCATTGCCCAGCTATCTGTCACTTCATCGAAAGGACAGTAGAAAAGGAAGGTGGCTCCTACAAATGCCA
TCATTGCGATAAAGGAAAGGCTATCATTCAAGATGCCTCTGCCGACAGTGGTCCCAAAGATGGACCCCCACCCACGAGGA
GCATCGTGGAAAAAGAAGACGTTCCAACCACGTCTTCAAAGCAAGTGGATTGATGTGACATCTCCACTGACGTAAGGGA
TGACGCACAATCCCACTATCCTTCGCAAGAC catttggagaggacagcccAAGCTTTCGGACCAGGCgtaTTCATCCCCC
gttgttgttgttatggtctaatttaaatatttaaatatggtctaaagaagaagaatTCGGACCAGGCgtaTTCATCCCCCGAATTCggtacgct
gaaatcaccag CCGAATTCGGTACGCTGAAATCACCAGTCTCTCTCTACAAATCTATCTCTCTCTATTTTCTCCATAAATAATGT
GTGAGTAGTTTCCCGATAAGGGAAATTAGGGTTCTTATAGGGTTTCGCTCATGTGTTGAGCATATAAGAAACCCTTAGTATG
TATTTGTATTTGTAAAATACTTCTATCAATAAAATTTCTAATTCCTAAAACCAAAATCCAGTACTAAAATCCAGATCTCCTAA
AGTCCCTATAGATCTTTGTCGTGAATATAAACCAGACACGAGACGACTAAACCTGGAGCCCAGACGCCGTTCGAAGCTAG
AAGTACCGCTTAGGCAGGAGGCCGTTAGGGAAAAGATGCTAAGGCAGGGTTGGTTACGTTGACTCCCCCGTAGGTTTGGT
```

Figure 6.10: Output of destination binary vector sequence after insertion of the designed STTM construct displayed by the DesignSTTM Web Portal.

6.4.3 Features

The features section of the output presents the integral list of features contained in the destination binary vector and their positions, as retrieved from the stored feature_list.txt file. The STTM sequence location is also displayed along with these features. Figure 6.11 shows a sample list of features in the pFGC5941 destination binary vector as displayed by the DesignSTTM Web portal.



Features List

GENE NAME	START	END	ORIENTATION
STTM	2252	13832	-
CDS-BlpR	376	924	plus
CDS-pVS1_RepA	7554	8622	plus
CDS-pVS1_StaA	6492	7121	plus
ORI-pBR322	9537	10156	minus
PRO-CaMVd35S	1543	2298	plus
PRO-Sp6	3172	3188	minus
PRO-T7	1505	1523	plus
REP-lacZ_a	4948	5104	plus
SEL-CAT	3253	3912	minus
SEL-kan1	10227	11021	minus

Figure 6.11 The output list of features present in the destination binary vector sequence after insertion of the designed STTM structure. These features are identified by the auto-annotation system embedded in the DesignSTTM Web Portal. The auto-annotation system comprises a database that stores the collected components (genes, promoters, replication origins etc.) of DNA sequences in various plasmid vectors, and a sequence similarity analysis algorithm called Basic Local Alignment Search Tool (BLAST).

6.4.4 STTM_Sequence

This section displays the STTM sequence and the different ways to incorporate the prepared STTM sequence into the binary vector, which include back-to-back primer pairing, face-to-face primer pairing, and oligo annealing. The back-to-back and face-to-face primer pairing methods provide the forward and reverse primers that can be used to amplify the STTM sequence using Polymerase chain reaction (**PCR**) technology, and then incorporate it into the binary vector, whereas oligo annealing is to synthesize the two strings—top and bottom strands of DNA, with addition of adaptors harboring the restriction enzyme site(s) selected (like PacI). The two strands synthesized can be dissolved in water and mixed in equal amounts before they are heated to a high temperature and then cooled down, during which the two strands anneal to produce double strands with two sticky-ends. This double strand DNA with two sticky-ends can be cut with restriction enzymes (like Pac I) and incorporated directly into the pre-digested vector.

6.4.5 E-Mail Report

The system combines all the results obtained and displayed to a user to formulate a PDF document report. The system sends this report as an attachment to the user's email.

To dynamically generate PDF reports, a TCPDF [21] PHP library is used. The PDF is prepared simultaneously when every section of the output is computed. For example, when the portal generates the plasmid map, it is written immediately to the PDF, and the same applies to the Sequence, Features, and the STTM-Sequence results as well.

A user can choose from two options to receive the generated PDF report: send to registered email or send to a different email. If the user selects the "Send to registered Email", the PDF prepared is written to a file with a name of "Merged_Report_STTM_< the miRNA name given > and store it in the personal directory of the user. The system will fetch the user's registered Email ID from the database, and then execute SendEmail program. If the user selects "Send to another mail" option, the system alerts a text field to get the Email ID from the user. The system validates the email address entered before it is used by the SendEmail program to send the PDF report to it.

6.5 Material_STTM

The main objective of the Material_STTM Web Portal is to support the distribution of STTM constructs, genotypic evidence and phenotypic information as well as the seeds of transgenic lines generated with the designed STTM constructs. The portal is designed to collect and show the availability of both STTM constructs of different miRNAs, the genotypic evidence, and the phenotypic alternations of transgenic crops lines transformed with previously designed STTM constructs. Material_STTM Web Portal is intended to serve as a venue through which various STTM plasmid vectors and transgenic seeds of various STTM constructs can be distributed to the research community. The contents of Material_STTM are displayed through a web portal called "STTM_Vectors and Transgenic Lines." Due to the unavailability of STTM vectors, the web portal of STTM_VECTORS_TRANSGENIC_LINES has not been constructed.

6.5.1 Web Portal of Transgenic Lines

The Web Portal of Transgenic Lines display page contains a list of transgenic lines of various STTM constructs. For each STTM transgenic line, there is various genotypic and phenotypic information that includes the following:

- Genotyping by sequencing: primers used to amplify genomic DNA in each of the transgenic lines to verify integration of STTM construct through regular PCR are provided along with cloned PCR products that are sequenced with the Sanger sequencing technology.
- Genotyping by qPCR: an integrated image is generated to show the expression levels of the following genes: a STTM, the STTM targeted microRNAs, and the target genes of the STTM-targeted microRNA.
- Genotyping by Northern: an integrated hybridization image produced from Northern Blot to show the expression levels of not only the STTM targeted miRNA but also the target genes of STTM targeted miRNA using Northern Blot technique.
- Phenotyping: images and plots to depict the altered phenotypes of the STTM transgenic lines in which the targeted miRNA's are destroyed.

This genotypic evidence is to confirm the transgenic lines of a STTM construct are genuine and the phenotypic alternations of transgenic lines of a STTM construct are caused by the degradation of STTM targeted miRNA's in transgenics.

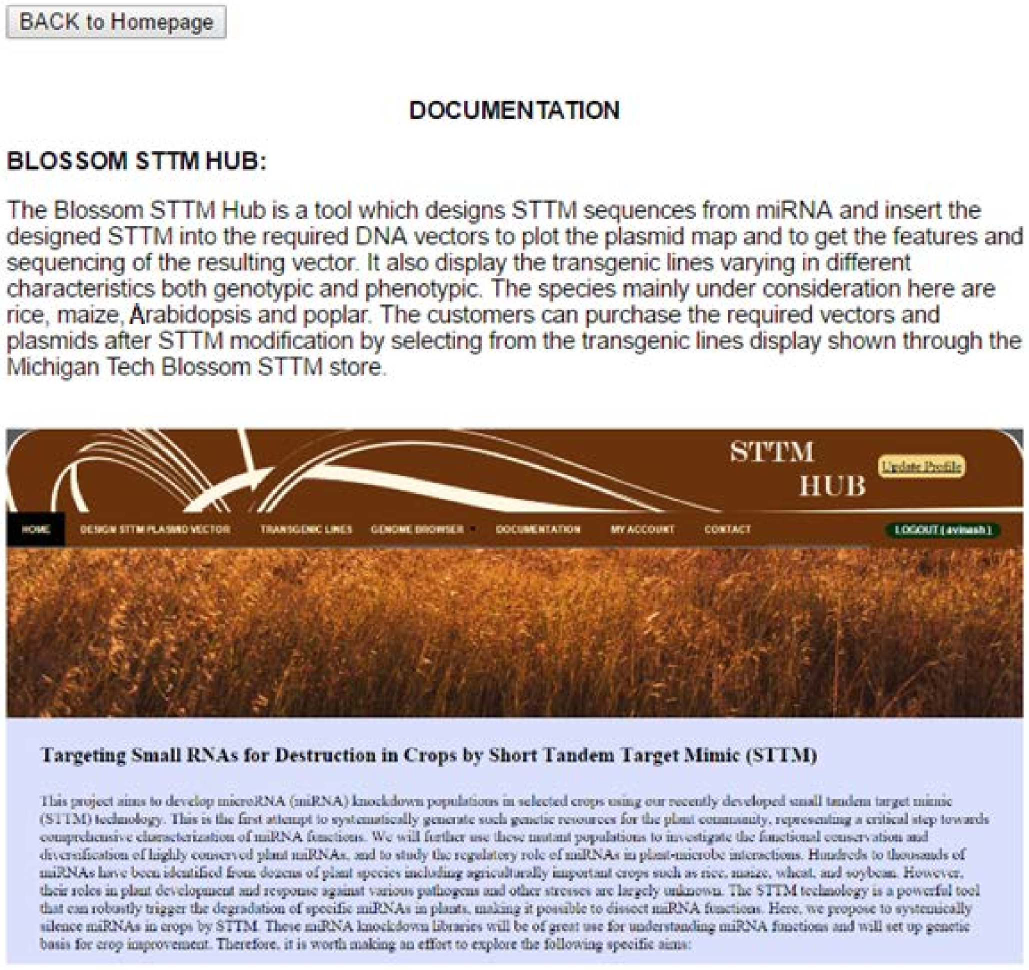
A link to the Blossom STTM E-store is provided along with every STTM construct, and a user can use each link to navigate to the store where they can purchase the STTM construct and the transgenic seeds from the STTM line they choose.

6.5.2 Blossom STTM E-Store

The Blossom STTM E-store was set up using the Michigan Tech Touchnet Marketplace Service. The cart functionality is integrated, and a user can select the required constructs and/or transgenic lines of a specific miRNA in a species, and then place an order for desired quantity. The system will record each order and assign each a transaction ID for tracking. The order will be received and processed within the specified period subject to availability [22]. The E-Store has a working prototype developed and can become existent in the market when more vectors and STTM constructs are added to the inventory upon further research.

6.6 Documentation

Documentation is very vital to a Web portal, as it acts as a reference for users on how to use the application, and provides directions for users to realize the different available features in the tool. A PDF document is made available to users through the Blossom STTM Web portal that contains all the critical steps involved in using the DesignSTTM and MaterialSTTM Web portals. Figure 6.12 shows the documentation file of the blossom STTM Hub.



DOCUMENTATION

BLOSSOM STTM HUB:

The Blossom STTM Hub is a tool which designs STTM sequences from miRNA and insert the designed STTM into the required DNA vectors to plot the plasmid map and to get the features and sequencing of the resulting vector. It also display the transgenic lines varying in different characteristics both genotypic and phenotypic. The species mainly under consideration here are rice, maize, Arabidopsis and poplar. The customers can purchase the required vectors and plasmids after STTM modification by selecting from the transgenic lines display shown through the Michigan Tech Blossom STTM store.

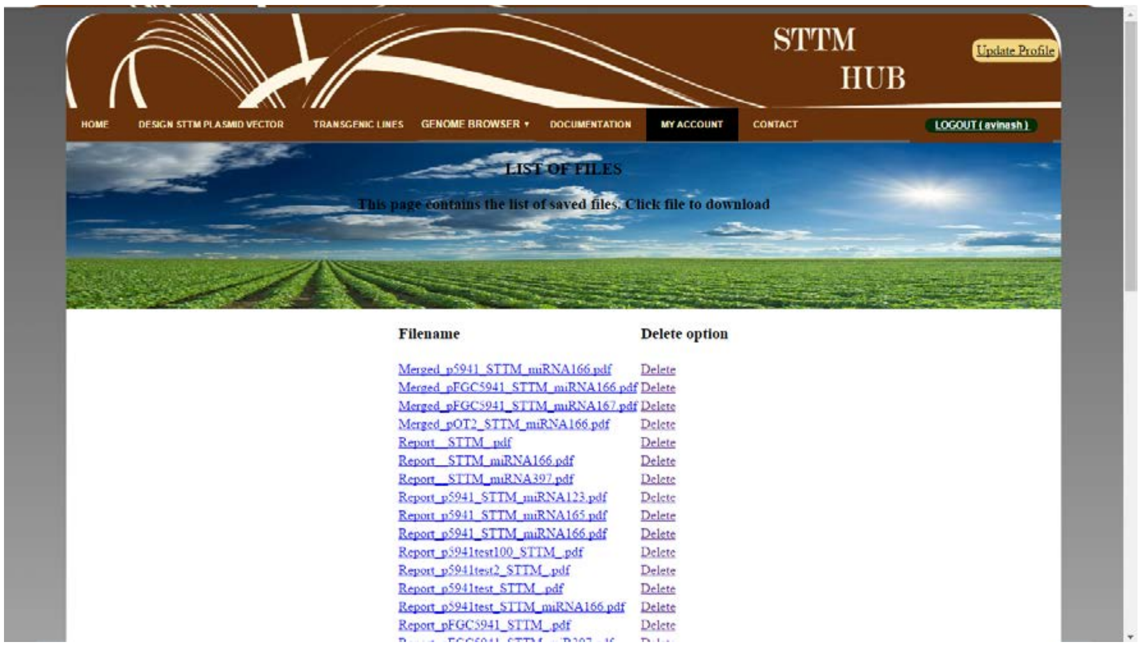
Targeting Small RNAs for Destruction in Crops by Short Tandem Target Mimic (STTM)

This project aims to develop microRNA (miRNA) knockdown populations in selected crops using our recently developed small tandem target mimic (STTM) technology. This is the first attempt to systematically generate such genetic resources for the plant community, representing a critical step towards comprehensive characterization of miRNA functions. We will further use these mutant populations to investigate the functional conservation and diversification of highly conserved plant miRNAs, and to study the regulatory role of miRNAs in plant-microbe interactions. Hundreds to thousands of miRNAs have been identified from dozens of plant species including agriculturally important crops such as rice, maize, wheat, and soybean. However, their roles in plant development and response against various pathogens and other stresses are largely unknown. The STTM technology is a powerful tool that can robustly trigger the degradation of specific miRNAs in plants, making it possible to dissect miRNA functions. Here, we propose to systemically silence miRNAs in crops by STTM. These miRNA knockdown libraries will be of great use for understanding miRNA functions and will set up genetic basis for crop improvement. Therefore, it is worth making an effort to explore the following specific aims:

Figure 6.12: Blossom STTM Hub Documentation file, which provides an instruction for using and understanding the DesignSTTM and MaterialSTTM Web Portals. The documentation can be reached from the Top Menu of the Blossom STTM Web Portal's interface.

6.7 User Account Web Interface

As aforementioned, a user has a personal directory created upon registration. The personal directory is a place where all results generated by the user are stored. A front-end interface called "My Account" was developed to display the list of all files stored in the user's personal directory. The user can browse the list of files stored in his personal directory, which is updated every time after the user utilizes the Blossom STTM Hub tool to design a STTM construct. The user can also download or delete any of these stored results through the "My Account" web interface. Figure 6.13 shows the front-end interface of the user's "My Account" section.



LIST OF FILES

This page contains the list of saved files. Click file to download

Filename	Delete option
Merged_p5941_STTM_miRNA166.pdf	Delete
Merged_pFGC5941_STTM_miRNA166.pdf	Delete
Merged_pFGC5941_STTM_miRNA167.pdf	Delete
Merged_pOT2_STTM_miRNA166.pdf	Delete
Report__STTM_.pdf	Delete
Report__STTM_miRNA166.pdf	Delete
Report__STTM_miRNA397.pdf	Delete
Report_p5941_STTM_miRNA123.pdf	Delete
Report_p5941_STTM_miRNA165.pdf	Delete
Report_p5941_STTM_miRNA166.pdf	Delete
Report_p5941test100_STTM_.pdf	Delete
Report_p5941test2_STTM_.pdf	Delete
Report_p5941test_STTM_.pdf	Delete
Report_p5941test_STTM_miRNA166.pdf	Delete
Report_pFGC5941_STTM_.pdf	Delete

Figure 6.13: The web interface of a user's account where a list of files stored in the user's personal directory. An option for downloading or deleting the files is provided. The files can be downloaded by clicking on the filename hyperlink.

7 TF Cluster: Web Portal

This section is independent from the STTM Web Portal. TF Cluster Web Portal is to develop an online version of TF-Cluster, which is a tool for building collaborative network of TFs and then decomposing it into collaborative sub-networks. Each sub-network has been proven to contain a group of TFs that are highly collaborative in controlling a biological process or a complex trait [23]. This section includes the following contents:

- Introduction to TF Cluster Web Portal.
- Architecture to TF Cluster Web Portal.
- Design of GNETINDEX Input Form.
- TF Cluster User File System.

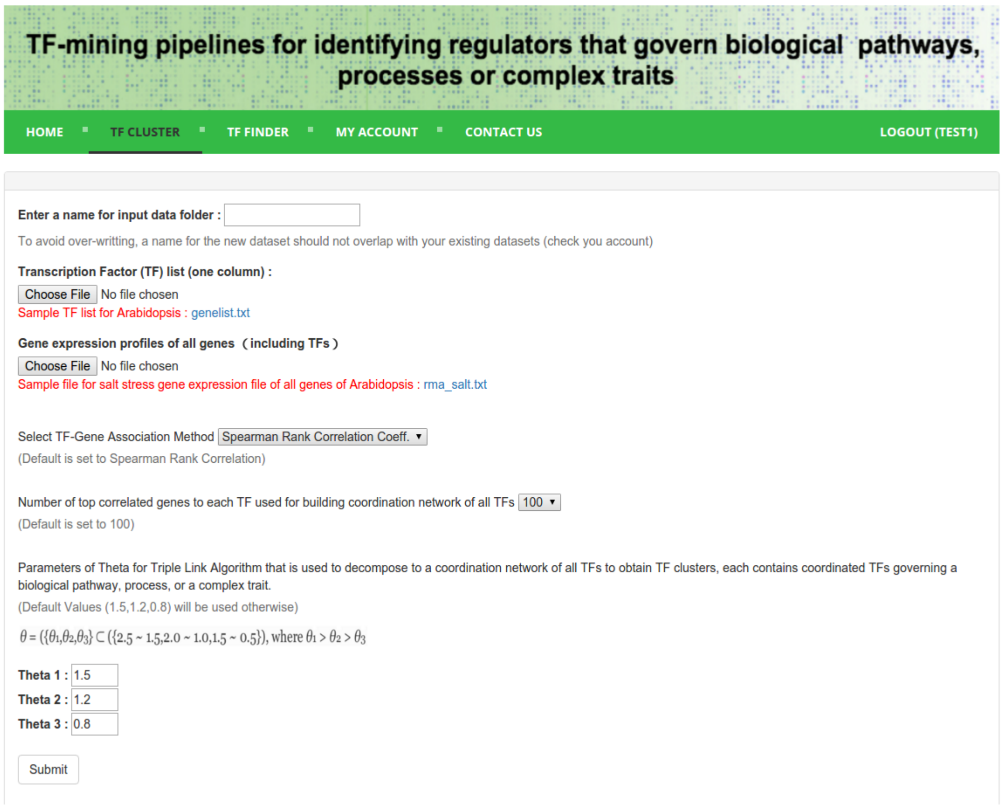
7.1 Introduction to TF Cluster Web Portal

Identification of key transcription factors (TFs) that control a biological process, pathway, or complex trait is an important step towards understanding of molecular regulatory mechanisms, but due to the presence of a large number of genes and sophisticated interaction among them, identifying these TF's becomes very challenging. The TF-Cluster Web portal can identify coordination TF clusters through construction of a coordination network of all TFs followed by network decomposition. Each resulting cluster contains a group of coordinated TFs known to control the same biological pathway, process, or complex trait. The TF-Cluster Web portal will facilitate the use of this pipeline to more gene expression data for novel biological knowledge discovery. The TF-Cluster interface was built with HTML. CSS was the front-end technologies whereas PHP and PERL were used as backend scripting technologies. The reason for developing the TF-Cluster as a Web application is to facilitate the use of the TF-Cluster for knowledge discovery. Additionally, the shortcomings of using standalone software and the complication ensues are eliminated in an online Web application.

7.2 Architecture to TF Cluster Web Portal

The TF-Cluster Web portal follows the renowned MVC (Model-View-Controller) architecture, which perfectly supports this application. The internal algorithm built using PERL and R programming forms the Model of the TF-Cluster architecture. The HTML, CSS, and JavaScript built Web portal forms the user interface of the application acting as the Controller. The presentation layer that stores the input data coming in temporary outputs, and user file systems with the outputs forms the presentation or the View layer of the system. The Web interface has been designed to represent a simple and responsive system that efficiently caters to the needs of a user to traverse through the portal with ease. The latest version of HTML and HTML5 was used along with CSS3 to build the client side pages. JavaScript was used to implement the dynamic functions needed in the system, and PHP was used to carry the input data submitted by the user to the back-end server for processing, and then to the TF-Cluster implementation algorithm, which executes the Perl and R programming scripts to create the outputs. The system then sends the outputs back to the Controller and issues the user an email with the results.

7.3 Design of GNETINDEX Input Form


TF-mining pipelines for identifying regulators that govern biological pathways, processes or complex traits
HOME
TF CLUSTER
TF FINDER
MY ACCOUNT
CONTACT US
LOGOUT (TEST1)
Choose File
Choose File
Submit

Enter a name for input data folder :

To avoid over-writting, a name for the new dataset should not overlap with your existing datasets (check you account)

Transcription Factor (TF) list (one column) :

No file chosen

Sample TF list for Arabidopsis : genelist.txt

Gene expression profiles of all genes （including TFs）

No file chosen

Sample file for salt stress gene expression file of all genes of Arabidopsis : rma_salt.txt

Select TF-Gene Association Method: Spearman Rank Correlation Coeff.

(Default is set to Spearman Rank Correlation)

Number of top correlated genes to each TF used for building coordination network of all TFs: 100

(Default is set to 100)

Parameters of Theta for Triple Link Algorithm that is used to decompose to a coordination network of all TFs to obtain TF clusters, each contains coordinated TFs governing a biological pathway, process, or a complex trait.

(Default Values (1.5,1.2,0.8) will be used otherwise)

$\theta = (\{\theta_1,\theta_2,\theta_3\} \subset (\{2.5 \sim 1.5, 2.0 \sim 1.0, 1.5 \sim 0.5\})$, where $\theta_1 > \theta_2 > \theta_3$

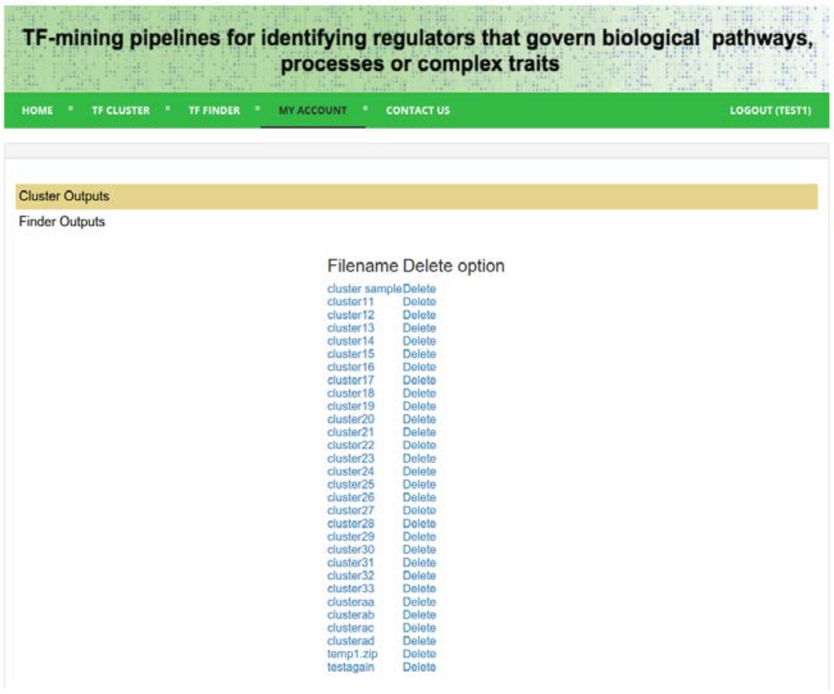
Theta 1 : 1.5
Theta 2 : 1.2
Theta 3 : 0.8

Figure 7.1: Depicts the TF-Cluster Web Portal Homepage. It is a web portal that accepts Transcription Factor (TF) list, Gene Expression Profiles, and the Theta values to build the collaborative TF network and then decompose it to acquire a number of sub-networks of collaborative TFs, each sub-network regulates a complex trait.

A user first enters the name for the data folder, which is where the output files would be stored in the file system after the execution of the TF-Cluster pipeline. The name should not be similar to any other data folder names they have used previously, and if they enter the same name, the system alerts the user with a message requesting them to choose a unique name for the data folder. The user then needs to submit the Transcription Factor list and the Gene Expression profiles of all the genes. Both files must be text files. The system provides sample files for each of the two input files next to each other, which would help the user to provide the correct input files. These two input files will be used to form the TF-gene correlation matrix which is a matrix of the form P * Q, where P is the number of TFs and Q is the number of genes. This matrix is to be computed by using one of the four correlation coefficients: Spearman Rank, Pearson, Weighted Rank, or Kendall correlation coefficients. The user chooses which of the four coefficients to use to compute the TF-gene correlation matrix. The next input is the drop down list from which the user can select how many correlated genes would be considered for each TF, which is used for building coordination networks of all TFs. The choices for this input can vary between 50, 100, 150, 200, 250, or 300, with 100 being the default value. Using the TF-gene correlation matrix, the Shared Co-expression Connectivity Matrix (SCCM) of P x P is built, in which each element of the matrix is the number of genes commonly represented in the top co-expressed genes to the two corresponding TFs in a row and a column. The TF-Cluster Web portal uses the Triple Link algorithm to decompose the SCCM matrix and to compute the final output. The algorithm searches through the entire SCCM matrix to identify the pair of TFs with the highest element and correlation. Once the pair of TFs—TF1 and TF2—is identified, the third TF3 would be added to the network only if it contains significant connectivity to the first two TFs, and the fourth TF4 is added to the network only if it contains at least three significant links to the first three TFs. The algorithm goes on to complete the closely correlated network of TFs and uses three parameters of "theta": $\theta1$, $\theta2$, and $\theta3$. The user is allowed to enter his own values of these θ based on his specific requirements.

Once the user successfully enters all inputs, the system validates the form for any errors before submitting to the backend where the input files are uploaded to the user file system in a separate folder created in the name specified by the user. The PHP script then executes the Triple link algorithm to get the output stored to the same folder as the input, which leads to easy identification for the user to locate his output. Once the user submits the form, a message showing successful submission is displayed, and the user will receive an email once the execution has been completed.

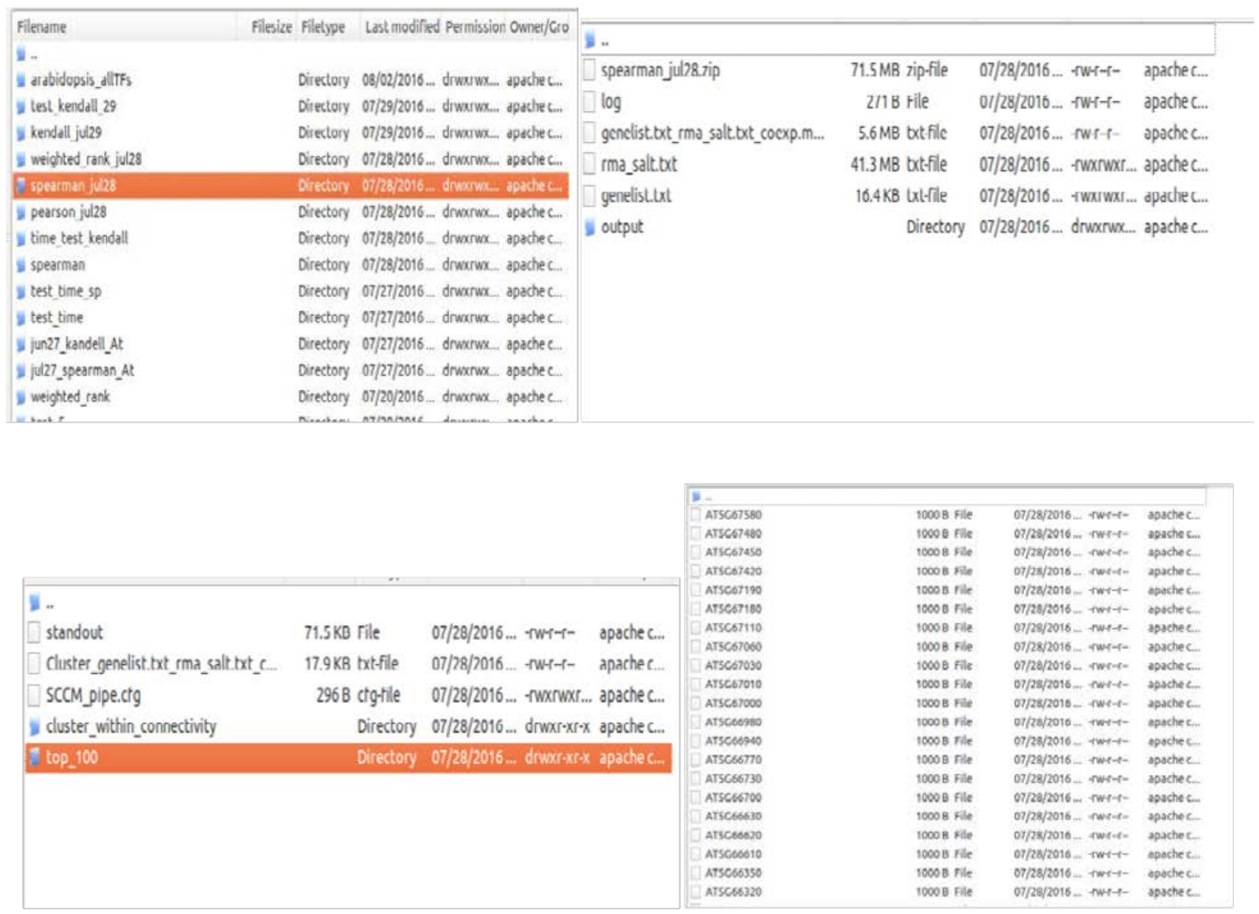
7.4 TF Cluster User File System


TF-mining pipelines for identifying regulators that govern biological pathways, processes or complex traits
HOME
TF CLUSTER
TF FINDER
MY ACCOUNT
CONTACT US
LOGOUT (TEST1)
Cluster Outputs
Finder Outputs

Filename	Delete option
cluster sample	Delete
cluster11	Delete
cluster12	Delete
cluster13	Delete
cluster14	Delete
cluster15	Delete
cluster16	Delete
cluster17	Delete
cluster18	Delete
cluster19	Delete
cluster20	Delete
cluster21	Delete
cluster22	Delete
cluster23	Delete
cluster24	Delete
cluster25	Delete
cluster26	Delete
cluster27	Delete
cluster28	Delete
cluster29	Delete
cluster30	Delete
cluster31	Delete
cluster32	Delete
cluster33	Delete
clusteraa	Delete
clusterab	Delete
clusterac	Delete
clusterad	Delete
temp1.zip	Delete
testagain	Delete

Figure 7.2: Illustration of the TF Cluster User File System Web Portal. It stores the results of TF Clusters derived from a high-throughput gene expression dataset by the TF Cluster Web Portal System. It is also a place users can download the resultant TF Clusters.

A user registered to the TF-Cluster Web portal will have his own file system under the My Account section. The user will be able to see all of his previous TF-cluster data folders and output files. The system provides each user the option of downloading these files whenever he wants to, and he can delete those files that are no longer needed. The user file system connects directly to the Apache server backend file system of the TF-Cluster Web portal. This makes the retrieval of data back from the server in the My Account section very efficient, minimizing the time taken for execution.



```
Filename                Filesize Filetype   Last modified   Permission Owner/Gro
..
arabidopsis_allTFs               Directory  08/02/2016...   drwxrwx... apache c...
test_kendall_29                  Directory  07/29/2016...   drwxrwx... apache c...
kendall_jul29                    Directory  07/29/2016...   drwxrwx... apache c...
weighted_rank_jul28              Directory  07/28/2016...   drwxrwx... apache c...
spearman_jul28                   Directory  07/28/2016...   drwxrwx... apache c...
pearson_jul28                    Directory  07/28/2016...   drwxrwx... apache c...
time_test_kendall                Directory  07/28/2016...   drwxrwx... apache c...
spearman                         Directory  07/28/2016...   drwxrwx... apache c...
test_time_sp                     Directory  07/27/2016...   drwxrwx... apache c...
test_time                        Directory  07/27/2016...   drwxrwx... apache c...
jun27_kandell_At                 Directory  07/27/2016...   drwxrwx... apache c...
jul27_spearman_At                Directory  07/27/2016...   drwxrwx... apache c...
weighted_rank                    Directory  07/20/2016...   drwxrwx... apache c...
```

```
..
spearman_jul28.zip                  71.5 MB zip-file   07/28/2016... -rw-r--r--  apache c...
log                                  271 B  File       07/28/2016... -rw-r--r--  apache c...
genelist.txt_rma_salt.txt_coexp.m... 5.6 MB txt-file   07/28/2016... -rw-r--r--  apache c...
rma_salt.txt                        41.3 MB txt-file   07/28/2016... -rwxrwxr... apache c...
genelist.txt                        16.4 KB txt-file   07/28/2016... -rwxrwxr... apache c...
output                                      Directory  07/28/2016... drwxrwx...  apache c...
```

```
..
standout                          71.5 KB File       07/28/2016... -rw-r--r--  apache c...
Cluster_genelist.txt_rma_salt.txt_c... 17.9 KB txt-file 07/28/2016... -rw-r--r--  apache c...
SCCM_pipe.cfg                      296 B  cfg-file   07/28/2016... -rwxrwxr... apache c...
cluster_within_connectivity              Directory  07/28/2016... drwxr-xr-x  apache c...
top_100                                  Directory  07/28/2016... drwxr-xr-x  apache c...
```

```
..
AT5G67580     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67480     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67450     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67420     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67190     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67180     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67110     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67060     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67030     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67010     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G67000     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66980     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66940     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66770     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66730     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66700     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66630     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66620     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66610     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66350     1000 B File   07/28/2016... -rw-r--r--   apache c...
AT5G66320     1000 B File   07/28/2016... -rw-r--r--   apache c...
```

Figure 7.3: Illustration of the TF Cluster result using the Triple Link Algorithm. It will contain the Top 100 TF’s present in the cluster from the input genes.

As mentioned before, every user has his own folder created at the time of registration to the TF-Cluster Web portal, and all the submissions he makes in the TF-cluster portal would be stored in separate folders inside the clusteroutput folder for all users. The above images depict the hierarchy of how the system creates and stores the output folders, and the outputs are stored dynamically inside specific folders. The data folder created for every submission would contain all the uploaded input files as well as the temporary files created by the Triple Link algorithm. Additionally, the system creates a folder named “output” for every submission where the result files are stored. This includes the Top 100 genes co-expressed as a TF computed in the end, and the folder is compressed and sent as an email attachment to the user’s email.

8 Conclusion

The DesignSTTM Web portal designs the STTM sequence using the miRNA sequence and the initial vector passed as inputs and the designed STTM sequence is targeted into the destination vector using PCR between one or two restriction enzymes chosen by a user. The user can use pOT2 or pFGC5941 as the initial vectors or can add any other vector of his choice as the initial vector. The resulting destination vector is plotted as a circular plasmid map. All the genes and enzymes present in the destination vector are identified by using BLAST tool and are legibly labeled using the automatic annotation system. This automatic annotation system makes use of the algorithms drawTextAlongArc() and draw_enzyme() to label all the genes, enzymes and other features present in plasmid. The circular plasmid map of the destination vector, containing the designed STTM sequence can be sent to the user’s email and also gets stored in the user’s personal directory. The MaterialSTTM Web portal represents genotypic and phenotypic evidence of STTM constructs in transgenic crops. Users can visually see evidence in transgenic lines of the designed STTM constructs. It also encompasses the portal to sell transgenic seeds to customers by taking in orders through the online store.

The TF-Cluster Web portal is an online tool that builds a collaborative network of Transcription Factors (TFs) and then decomposes it into highly collaborative subnetworks. This portal will greatly facilitate the use of the Triple Link algorithm for identifying the transcription factors (TFs) as different clusters or subnetworks from high throughput gene expression data. TFs in each cluster or subnetwork are found to govern a biological process or a complex trait in plants. The front-end user interface of this portal contains the input form called GNETINDEX and this form requests the list of TF's, gene expression profiles of all genes and one of the four correlation coefficients: Spearman Rank, Pearson, Weighted Rank, or Kendall correlation coefficients. A user provides all the inputs and chooses from the option of four correlation coefficients to use to compute the TF-gene correlation matrix. When the user submits the form, the portal uses the Triple Link Algorithm to show the top 100 subnetworks that are expressed by the TFs. The result is sent to the user in an email and can also be obtained from the user's personal directory in the server.

9 Future Work

As part of future enhancements to the Blossom STTM Hub, the gene repository supporting the transfer of STTM could be increased as more successful results are obtained. Additionally, more transgenic lines can be made available to users as researchers produce results working on the STTM technology. The algorithm used to plot the circular plasmid maps can be extended to serve other applications in which data visualization is needed for large volumes of data.

10 References

[1] J. Y. G. X. J. W. K. S. P. X. T. X. C. a. G. T. Yan, "Effective small RNA destruction by the expression of a short tandem target mimic in *Arabidopsis*," *The Plant Cell,* vol. 24, no. 2, pp. 415 - 427, 2012.

[2] X. D. N. F. W. Y. J. G. Y. T. X. .. &. T. G. Jia, "Functional plasticity of miR165/166 in plant development revealed by small tandem target mimic," *Plant Science,* vol. 233, pp. 11-21, 2015.

[3] D. P. Bartel, "MicroRNAs: genomics, biogenesis, mechanism, and function.," *Cell,* vol. 2, no. 116, pp. 281-297, 2004.

[4] G. J. Y. Y. G. M. Q. R. F. Y. M. a. X. T. Tang, "Construction of short tandem target mimic (STTM) to block the functions of plant and animal microRNAs," *Methods,* vol. 58, no. 2, pp. 118 - 125, 2012.

[5] X. P. S. I. J. F. a. D. S. W. Dong, "PlasMapper: a web server for drawing and auto-annotating plasmid maps," *Nuclear acids research,* p. 5, 2004.

[6] S. F. W. G. W. M. E. W. M. a. D. J. L. Altschul, "Basic local alignment search tool," *Journal of molecular biology ,* vol. 215, no. 3, pp. 403-410, 1990.

[7] H. E. a. D. L. Williams, Web database applications with PHP and MySQL, O'Reilly Media, 2004.

[8] D. Powers, PHP solutions: dynamic web design made easy, Apress, 2014.

[9] P. B. A. a. F. S. Lubbers, Overview of HTML5, In Pro HTML5 Programming, Apress, 2011.

[10] P. M. A. V. a. D. L.-d.-I. Garaizar, "Presentation accuracy of the Web revisited: animation methods in the HTML5 era," *PloS one 9,* vol. 10, no. e109812, 2014.

[11] M. Grady, "Functional programming using JavaScript and the HTML5 canvas element," *Journal of computing sciences in colleges,* vol. 26, no. 2, pp. 97 - 105, 2010.

[12] C. B. J. O. S. a. L. H. Lienert, "Current trends in vector-based Internet mapping: A technical review," *InOnline maps with APIs and WebServices,* no. Springer Berlin Heidelberg, pp. 23-36, 2012.

[13] J. a. K. S. Chaffer, Learning jquery: better interaction design and web development with simple javascript techniques, Packt Publishing, 2007.

[14] V. T. P. J. M. D. Roberts RJ, "REBASE: restriction enzymes and methyltransferases.," *Nucleic Acids Research,* vol. 1, no. 31, pp. 418-420, 2003.

[15] M. T., "The BLAST Sequence Analysis Tool - The NCBI Handbook [Internet] - 2nd Edition," 15 March 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK153387/.

[16] T. T. D. R. M. Kim D. Pruitt, "NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins," *Nucleic Acids Research,* 33(1), pp. 501-504, 2005.

[17] I. M. Y. a. J. B. Korf, Blast, O'Reilly Media, Inc, 2003.

[18] C. R. G. G. P. T. L. a. R. A. Györödi, "Web 2.0 Technologies with jQuery and Ajax," in *Computer Technology and Computer Programming: Research and Strategies*, 2011.

[19] K. B. F. F. a. R. A. G. Mullis, in *Polymerase Chain Reaction*, Boston, MA, 1994.

[20] D. Shappir, "Performing binary composition of images onto an html canvas element". US Patent U.S. Patent Application 13/414,735, 8 March 2012.

[21] V. Chowdhary*, Generate PDF with PHP: FPDF, TCPDF, DOMPDF, ezPDF, FPDI and HTML2PDF*, 2011.

[22] C. a. M. B. Darie, Beginning PHP 5 and MySQL E-Commerce: From Novice to Professional, Apress, 2004.

[23] J. e. a. Nie, "TF-Cluster: a pipeline for identifying functionally coordinated transcription factors via network decomposition of the shared coexpression connectivity matrix (SCCM)," *BMC Systems Biology 5,* 2011.

[24] M. Schrenk, Webbots, spiders, and screen scrapers: A guide to developing Internet agents with PHP/CURL, No Starch Press, 2012.

[25] T. Merz, "HTML and PDF, Web Publishing with Acrobat/PDF,," *Springer Berlin,* pp. 2 - 8, 1998.

[26] C. K. D. K. M. A. R. a. J. C. C. Llave, "Endogenous and silencing-associated small RNAs in plants," *The Plant Cell*, vol. no. 7, no. 14, pp. 1605 - 1619, 2002.

[27] W. L. H. L. L. a. J. L. Cui, "The research of PHP development framework based on MVC pattern," *Computer Sciences and Convergence Information Technology - Fourth International Conference on IEEE,* pp. 947-949, 2009.

[28] C. Russell, PHP development tool essentials, 2016.

[29] M. Thomas, "The BLAST Sequence Analysis Tool," 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK184580/?report=reader.

[30] W. Gilmore, "Creating Ajax-enhanced Features with jQuery and PHP," in *Beginning PHP and MySQL: From Novice to Professional*, 2010, pp. 437-448.

[31] G. J. Y. Y. G. M. Q. R. F. e. a. Tang, "Construction of short tandem target mimic (STTM) to block the functions of plant and animal microRNAs," *Methods,* no. 58, pp. 118-125, 2012.

APPENDIX A

SOURCE CODE – FUNCTION DEFINITION draw_arc()

```
function draw_arc(start,end,lnwidth,color,label_x_val,label_y_val,is_direction,comp)
{
        // Get value of the radius for the circular map from ASSIGNED_RADIUS global variable
and if it is not set, assign it to 250 which is the default radius value in pixels
         var radius = ASSIGNED_RADIUS;
         if(!(ASSIGNED_RADIUS))
                radius = 250;
    var counterClockwise = false;

        // Position of starting and ending positions of the feature to be labeled
         var temp1=(start/Math.PI)*((<?php echo json_encode($totalbp);?>)/2);
         var temp2=(end/Math.PI)*((<?php echo json_encode($totalbp);?>)/2);
         temp1=parseInt(temp1);
         temp2=parseInt(temp2);

        //Draw the arc between the starting and ending positions of the feature using context.arc()
         context.beginPath();
        context.arc(x, y, radius, (((temp1)/((<?php echo json_encode($totalbp);?>)/2))*Math.PI),
(((temp2)/((<?php echo json_encode($totalbp);?>)/2))*Math.PI), counterClockwise);
         context.lineWidth = lnwidth;
         context.strokeStyle = color;
         context.stroke();
         context.closePath();
         context.beginPath();

        //To draw the arrow in clockwise direction is is_direction is PLUS
         if((is_direction == "plus")&&((temp2-temp1)>50))
         {
          context.lineWidth = 10;
          context.strokeStyle = color;
          context.stroke();
          context.closePath();
          context.beginPath();
          context.lineWidth = 2;
          var arrow1_startx = getPointx(x,y,radius+15,(((temp2-(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
          var arrow1_starty = getPointy(x,y,radius-15,(((temp2-(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
          var arrow2_startx = getPointx(x,y,radius-15,(((temp2-(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
```

```
                var arrow2_starty = getPointy(x,y,radius-15,(((temp2-(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow3_startx = getPointx(x,y,radius,(((temp2+15)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow3_starty = getPointy(x,y,radius,(((temp2+15)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                context.moveTo(arrow1_startx,arrow1_starty);
                context.lineTo(arrow2_startx,arrow2_starty);
                context.lineTo(arrow3_startx,arrow3_starty);
                context.lineTo(arrow1_startx,arrow1_starty);
                context.fillStyle = color;
                context.fill();
                context.stroke();
                context.closePath();
               }

             //To draw the arrow in anti-clockwise direction is is_direction is MINUS
               if((is_direction == "minus")&&((temp2-temp1)>50))
               {
                context.arc(x,y,radius,(((temp1+(radius/7))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI),(((temp1+(radius/7)+60)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI),false);
                context.lineWidth = 10;
                context.strokeStyle = color;
                context.stroke();
                context.closePath();
                context.beginPath();
                context.lineWidth = 2;
                var arrow1_startx = getPointx(x,y,radius,(((temp1+(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow1_starty = getPointy(x,y,radius,(((temp1+(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow2_startx = getPointx(x,y,radius-15,(((temp1+(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow2_starty = getPointy(x,y,radius-15,(((temp1+(radius/5))/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow3_startx = getPointx(x,y,radius,(((temp1-15)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                var arrow3_starty = getPointy(x,y,radius,(((temp1-15)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI));
                context.moveTo(arrow1_startx,arrow1_starty);
                context.lineTo(arrow2_startx,arrow2_starty);
                context.lineTo(arrow3_startx,arrow3_starty);
                context.lineTo(arrow1_startx,arrow1_starty);
```

```
                  context.fillStyle = color;
                  context.fill();
                  context.stroke();
                  context.closePath();
                }
        }
```

APPENDIX B

SOURCE CODE – FUNCTION DEFINITION drawTextAlongArc ()

```
function drawTextAlongArc(context, str, centerX, centerY, radius, start, end) {

  var len = str.length, s;
  var startbp=parseInt((start/Math.PI)*((<?php echo json_encode($totalbp);?>)/2));
  var endbp=parseInt((end/Math.PI)*((<?php echo json_encode($totalbp);?>)/2));

//Check if the feature is in the top semicircle of the circular plasmid map
  if(((startbp)>(((parseInt(<?php echo
json_encode($totalbp);?>))/10)*2))&&((startbp)<(((parseInt(<?php echo
json_encode($totalbp);?>))/10)*7)))
  {

    // Check if the feature is atleast 5% of the total length of the circular plasmid - that is atleast
1/20th of the total circumference of the circular map
            if((endbp-startbp)>((parseInt(<?php echo json_encode($totalbp);?>))/20))
            {
      var totalbpval = (parseInt(<?php echo json_encode($totalbp);?>));
      if((endbp-startbp)>(totalbpval/10))
      {
        var mid = (endbp + startbp)/2;
        startbp = mid - (totalbpval/20);
        endbp = mid + (totalbpval/20);
      }
      context.save();
      context.translate(centerX, centerY);
      var multiplier = ((endbp)/(<?php echo json_encode($totalbp);?>))*2*Math.PI;
      var anglesize = 2*((endbp - startbp)/(<?php echo json_encode($totalbp);?>))
      var angle = -1*2*Math.PI*((endbp - startbp)/(<?php echo json_encode($totalbp);?>));
      context.rotate((multiplier+Math.PI));
      context.rotate(-1 * (angle / len) / 2);
      for(var n = 0; n < len; n++) {
        context.rotate((angle / len));
        context.save();
        context.translate(0, 1 * radius);
        s = str[n];
        context.font="bold 14px Calibri";
        context.textAlign = 'center';
        context.fillStyle = "black";
        context.fillText(s, 0, 0);
        context.restore();
      }
      context.restore();
      }
         }

    //If the feature is in the bottom semicircle of the circular plasmid map
    else
         {

           if((endbp-startbp)>((parseInt(<?php echo json_encode($totalbp);?>))/20))
           {
        var totalbpval = (parseInt(<?php echo json_encode($totalbp);?>));
        if((endbp-startbp)>(totalbpval/10))
        {
          var mid = (endbp + startbp)/2;
          startbp = mid - (totalbpval/20);
          endbp = mid + (totalbpval/20);
        }
        context.save();
        context.translate(centerX, centerY);
        var multiplier = ((startbp)/(<?php echo json_encode($totalbp);?>))*2*Math.PI;
        var anglesize = 2*((endbp - startbp)/(<?php echo json_encode($totalbp);?>))
        var angle = 2*Math.PI*((endbp - startbp)/(<?php echo json_encode($totalbp);?>));

        context.rotate(multiplier);
        context.rotate(-1 * (angle / len) / 2);
        for(var n = 0; n < len; n++) {
          context.rotate((angle / len));
          context.save();
          context.translate(0, -1 * radius);
          s = str[n];
          context.font="bold 14px Calibri";
          context.textAlign = 'center';
          context.fillStyle = "black";
          context.fillText(s, 0, 0);
          context.restore();
        }
        context.restore();
      }
    }

  }
```

APPENDIX C

SOURCE CODE – FUNCTION DEFINITION draw_enzyme_arc ()

```
function
draw_enzyme_arc(start,end,lnwidth,color,label_x_val,label_y_val,is_direction,comp,isgene)
{

            // Get value of the radius for the circular map from ASSIGNED_RADIUS global
variable and if it is not set, assign it to 250 which is the default radius value in pixels
        var radius = ASSIGNED_RADIUS;
        if(!(ASSIGNED_RADIUS))
                radius = 250;

   var counterClockwise = false;
        var startbp=parseInt((start/Math.PI)*((<?php echo json_encode($totalbp);?>)/2));
        var endbp=parseInt((end/Math.PI)*((<?php echo json_encode($totalbp);?>)/2));
        start = start - ((Math.PI)/2);
        end = end - ((Math.PI)/2);
        context.beginPath();
        context.arc(x, y, radius, start, end, counterClockwise);
        context.lineWidth = lnwidth+20;
        context.strokeStyle = color;
        var sttmname = <?php echo json_encode($sttm_name);?>;
        if(isgene == 1)
        {
                if(comp=="STTM_")
                {
                        context.fillStyle = "brown";
                        context.strokeStyle = "brown";
                        context.lineWidth = lnwidth+20;
                }
                else
                {
                        context.fillStyle = "red";
                        context.strokeStyle = "red";
                }
        }
        else
        {
                context.fillStyle = "black";
                context.strokeStyle = "black";
        }
   context.stroke();
        context.closePath();
        context.beginPath();
        var angle = (start+end)/2;
        var line1_startx = getPointx(x,y,(radius+35),angle);
        var line1_starty = getPointy(x,y,(radius+35),angle);

//These are conditions to check which octant the enzymes will fall in

var OCTANT1CONDITION = ((angle>(-0.50)*(Math.PI))&&(angle<((-0.25)*(Math.PI)/2)));
var OCTANT2CONDITION = ((angle>(-0.25)*(Math.PI))&&(angle<0));
var OCTANT3CONDITION = ((angle>0)&&(angle<(0.25*(Math.PI))));
var OCTANT4CONDITION = ((angle>(0.25)*(Math.PI))&&(angle<((0.50)*(Math.PI))));
var OCTANT5CONDITION = ((angle>(0.50)*(Math.PI))&&(angle<((0.75)*(Math.PI))));
var OCTANT6CONDITION = ((angle>(0.75)*(Math.PI))&&(angle<(1*(Math.PI))));
var OCTANT7CONDITION = ((angle>(-1)*(Math.PI))&&(angle<((-0.75)*(Math.PI)/2)));
var OCTANT8CONDITION = ((angle>(-0.75)*(Math.PI))&&(angle<((-0.50)*(Math.PI)/2)));

//Use this method to label enzymes in 1st Octant
if((OCTANT1CONDITION))
{
        movex1 = 50;
        movey1 = movey1 + 15;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx+movex1;
        line1_starty = line1_starty+movey1;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="left";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx+5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx+5, line1_starty+10);
        }
}
//Use this method to label enzymes in 2nd Octant
else if((OCTANT2CONDITION))
{
        movex2 = 50;
        movey2 = movey2 + 15;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx-movex2;
        line1_starty = line1_starty-movey2;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="right";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx-5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx-5, line1_starty+10);
        }
}
//Use this method to label enzymes in 3rd Octant
else if((OCTANT3CONDITION))
{
        movex3 = 50;
        movey3 = movey3 + 15;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx-movex3;
        line1_starty = line1_starty-movey3;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="right";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx-5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx-5, line1_starty+10);
        }
}

//Use this method to label enzymes in 4th Octant
else if((OCTANT4CONDITION))
{
        movey4 = movey4 + 13;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx+movex4+10;
        line1_starty = line1_starty+movey4-5;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="left";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx+5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx+5, line1_starty+10);
        }
}
//Use this method to label enzymes in 5th Octant
else if((OCTANT5CONDITION))
{
        movey5 = movey5 + 13;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx+movex5;
        line1_starty = line1_starty+movey5;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="left";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx+5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx+5, line1_starty+10);
        }
}

//Use this method to label enzymes in 6th Octant
else if((OCTANT6CONDITION))
{
        movey6 = movey6 + 13;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx+movex6;
        line1_starty = line1_starty+movey6;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="left";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx+5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx+5, line1_starty+10);
        }
}

//Use this method to label enzymes in 7th Octant
else if((OCTANT7CONDITION))
{
        movex7 = 50;
        movey7 = movey2 + 15;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx-movex2;
        line1_starty = line1_starty-movey2;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
        context.textAlign="right";
        if(comp==="STTM_")
        {
                context.fillText(comp+sttmname, line1_startx-5, line1_starty+10);
        }
        else
        {
                context.fillText(comp+"   "+startbp, line1_startx-5, line1_starty+10);
        }
}
//Use this method to label enzymes in 8th Octant
else
{
        movex8 = 50;
        movey8 = movey2 - 15;
        context.moveTo(line1_startx,line1_starty);
        line1_startx = line1_startx+movex1;
        line1_starty = line1_starty+movey1;
        context.lineTo(line1_startx,line1_starty);
        context.moveTo(line1_startx,line1_starty);
        context.font = 'italic 15pt Calibri';
```

```
          context.textAlign="left";
          if(comp==="STTM_")
          {
                    context.fillText(comp+sttmname, line1_startx+5, line1_starty+10);
          }
          else
          {
                    context.fillText(comp+"   "+startbp, line1_startx+5, line1_starty+10);
          }

}

          context.lineWidth = 2;
   context.stroke();
          context.closePath();
          context.beginPath();



//If the enzyme orientation is PLUS, i.e. if equal to 1, mark the enzyme at the 5 prime end of the
enzyme which is the starting position of the enzyme and if it is MINUS, vice versa
          if(is_direction == 1)
          {
                    context.arc(x,y,radius,(((temp2-60)/((<?php echo
          json_encode($totalbp);?>)/2))*Math.PI),end,false);
                    context.lineWidth = 15;
                    context.strokeStyle = "black";
                    context.stroke();
                    context.closePath();
                    context.beginPath();
                    context.lineWidth = 2;
                    var circle_angle=((temp2/((<?php echo json_encode($totalbp);?>)/2))*Math.PI);
                    var circle_startx = getPointx(x,y,radius,end);
                    var circle_starty = getPointy(x,y,radius,end);
                    var direction_angle = end-Math.PI;
                    context.arc(circle_startx,circle_starty, 15,(Math.PI+direction_angle),((2 *
Math.PI)+direction_angle), false);
                    context.fill();
                    context.stroke();
                    context.closePath();
          }
          if(is_direction == 2)
          {
                    context.arc(x,y,radius,start,(((temp1+60)/((<?php echo
json_encode($totalbp);?>)/2))*Math.PI),false);
```

```
			context.lineWidth = 15;
			context.strokeStyle = "black";
			context.stroke();
			context.closePath();
			context.beginPath();
			context.lineWidth = 2;

			var circle_angle=((temp2/((<?php echo json_encode($totalbp);?>)/2))*Math.PI);
			var circle_startx = getPointx(x,y,radius,start);
			var circle_starty = getPointy(x,y,radius,start);
			var direction_angle = start-Math.PI;
			context.arc(circle_startx,circle_starty, 15,(direction_angle),(Math.PI + 
direction_angle), false);
			context.fill();
			context.stroke();
			context.closePath();
		}
}
```