CHAPTER 4

Methodology
4.1 Development of software for combining mass spectrometry data

4.1.1 Development of web-interfaces for searching in X! Tandem and OMSSA

For development of web-interface, CSS based HTML design was created using Macromedia Dreamweaver software. Zope web application server (Version 2.9.0) was used as the front-end. Native JavaScript and JQuery (Version v.1.3.0) library were used for client-side scripting. Box 1 describes the technical information of the web-application server MS-Analysis Suite. A secure login is available on the homepage and users need a register initially to get a login. Upon

![Box 1: Technical Information of MS-Analysis Suite App](image)

**Application Server:** Zope 2.9.8  
**Client-Side Scripting:** JQuery v.1.3.0, JavaScript  
**Server-Side Scripting:** Python 2.4  
**Database:** MySQL 5.0.22  
**Python DB API-2.0 interface:** MySQLdb  
**Graph Chart:** Open Flash Chart 1.9.7  
**Python Graphics:** Python Imaging Library (PIL) 1.1.6

![Figure 10: A screenshot of the query page of X! Tandem](image)
verification of the credential, the users can access the MS-Analysis Suite App Server. A separate web-interface has been created for searching in X! Tandem and OMSSA. Figure 10 and 11 describes a screenshot of the query page of X! Tandem and OMSSA, respectively. Once, a search is submitted, a search thread is created and a unique search ID will be created and the page is refreshed every 60 seconds to check if the results are available. Alternatively, the status can also be accessed through the search log. Once, the search is completed, user is redirected to the protein and peptide summary page (Figure 12) which gives a detailed view of the peptides and proteins identified. The peptide sequence is hyperlinked to the spectrum view page (Figure 13). This page lists the spectrum and peptide details along with an annotated graphical stick diagram of the MS/MS spectra along with a table describing the entire list of fragment ions and also highlights the matched fragment ions (Figure 14).

Figure 11: A screenshot of the query page of OMSSA.
Figure 12: A screenshot of the results summary page of X1 Tandem.

Figure 13: A screenshot of the spectrum view of annotated MS/MS spectra.
<table>
<thead>
<tr>
<th>#</th>
<th>+1k</th>
<th>+1k 17</th>
<th>+1k 18</th>
<th>seq</th>
<th>+1p</th>
<th>+1p 17</th>
<th>+1p 18</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>88.040</td>
<td>71.013</td>
<td>70.029</td>
<td>S</td>
<td>2987.213</td>
<td>2970.396</td>
<td>2969.352</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>275.099</td>
<td>206.072</td>
<td>207.086</td>
<td>H</td>
<td>2987.213</td>
<td>2970.396</td>
<td>2969.352</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>308.108</td>
<td>311.091</td>
<td>310.097</td>
<td>C</td>
<td>2750.284</td>
<td>2733.228</td>
<td>2732.244</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>441.192</td>
<td>424.165</td>
<td>423.181</td>
<td>I</td>
<td>2547.245</td>
<td>2530.218</td>
<td>2529.234</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>569.291</td>
<td>552.224</td>
<td>551.240</td>
<td>A</td>
<td>2547.245</td>
<td>2530.218</td>
<td>2529.234</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>646.291</td>
<td>681.267</td>
<td>680.283</td>
<td>E</td>
<td>2406.102</td>
<td>2389.076</td>
<td>2389.072</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>797.591</td>
<td>780.329</td>
<td>779.351</td>
<td>V</td>
<td>2277.060</td>
<td>2260.033</td>
<td>2259.049</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>926.404</td>
<td>909.378</td>
<td>908.394</td>
<td>E</td>
<td>2177.991</td>
<td>2160.905</td>
<td>2159.981</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>1040.447</td>
<td>1022.421</td>
<td>1022.437</td>
<td>N</td>
<td>2049.494</td>
<td>2032.422</td>
<td>2030.930</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>1151.474</td>
<td>1135.440</td>
<td>1137.464</td>
<td>D</td>
<td>1934.908</td>
<td>1917.879</td>
<td>1916.898</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>1293.517</td>
<td>1287.490</td>
<td>1286.506</td>
<td>E</td>
<td>1819.679</td>
<td>1802.652</td>
<td>1801.669</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>1415.557</td>
<td>1399.531</td>
<td>1397.547</td>
<td>M</td>
<td>1690.836</td>
<td>1673.810</td>
<td>1672.826</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>1515.610</td>
<td>1495.583</td>
<td>1494.596</td>
<td>P</td>
<td>1579.796</td>
<td>1562.760</td>
<td>1561.785</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>1592.647</td>
<td>1586.620</td>
<td>1585.636</td>
<td>A</td>
<td>1482.743</td>
<td>1465.716</td>
<td>1464.732</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>1690.674</td>
<td>1684.647</td>
<td>1680.663</td>
<td>D</td>
<td>1391.706</td>
<td>1374.675</td>
<td>1373.693</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>1811.758</td>
<td>1795.731</td>
<td>1793.747</td>
<td>L</td>
<td>1278.679</td>
<td>1261.652</td>
<td>1259.668</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>1908.811</td>
<td>1893.784</td>
<td>1890.800</td>
<td>P</td>
<td>1163.595</td>
<td>1146.556</td>
<td>1145.574</td>
<td>11</td>
</tr>
<tr>
<td>19</td>
<td>2109.927</td>
<td>2093.900</td>
<td>2086.918</td>
<td>L</td>
<td>979.512</td>
<td>962.483</td>
<td>961.499</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>2179.964</td>
<td>2162.937</td>
<td>2161.953</td>
<td>A</td>
<td>866.426</td>
<td>849.399</td>
<td>848.415</td>
<td>8</td>
</tr>
<tr>
<td>21</td>
<td>2251.001</td>
<td>2235.975</td>
<td>2232.991</td>
<td>A</td>
<td>793.399</td>
<td>776.362</td>
<td>775.379</td>
<td>7</td>
</tr>
<tr>
<td>22</td>
<td>2366.029</td>
<td>2349.001</td>
<td>2346.017</td>
<td>D</td>
<td>724.352</td>
<td>707.325</td>
<td>706.341</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>2513.096</td>
<td>2496.070</td>
<td>2495.086</td>
<td>F</td>
<td>609.325</td>
<td>592.298</td>
<td>591.314</td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>2612.165</td>
<td>2595.139</td>
<td>2594.154</td>
<td>V</td>
<td>467.256</td>
<td>445.230</td>
<td>444.246</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>2741.207</td>
<td>2724.181</td>
<td>2723.197</td>
<td>E</td>
<td>363.188</td>
<td>346.181</td>
<td>345.211</td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td>2828.239</td>
<td>2811.213</td>
<td>2810.229</td>
<td>S</td>
<td>234.145</td>
<td>217.119</td>
<td>216.152</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>2869.302</td>
<td>2852.276</td>
<td>2851.292</td>
<td>X</td>
<td>147.113</td>
<td>130.087</td>
<td>129.103</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 14:** A screenshot of the theoretical fragment ions of the identified peptide. Matched fragments ions are highlighted in blue, green, red and purple color.
4.1.2 Development of parsers for search algorithms

The search algorithms for interpretation of mass spectrometry derived data requires MS/MS data in various formats including mascot generic format (.mgf), peak list format (.pkl) and .dta files. Figure 15 describes the mascot generic format in detail. The search results derived from these search algorithms are in various formats. This include .dat format for Mascot, XML format for X! Tandem (.xml) and OMSSA (.omx). The first step in developing software for combining mass spectrometry data is to develop software programs for parsing these search results.

![Mascot Generic Formats (.mgf)](image)

**Figure 15:** Examples of the .mgf peak list used when submitting the tandem MS data to the algorithms. The charge state of the peptide ions from which the MS/MS spectra are derived may be known or unknown.
The XML output file of X! Tandem is based on the Biological Markup Language (BIOML) and contains the peptide, protein and spectrum information for a given search along with the search parameters. Figure 16 shows the screenshot of a sample XML output file from X! Tandem. The various parameters for a MS/MS spectrum result include spectrum_id, log e-value, log i-value, mh, protein_id, protein expect value, protein_uid, protein name, protein sumI, peptide_id, peptide start, peptide end, peptide mh, peptide delta, peptide hyperscore, peptide sequence, peptide missed cleavages, modifications, pkl name and charge state. The XML output file (.omx file) from OMSSA is generic XML file and contains a XML schema definition. Figure 17 describes a screenshot of a sample .omx file. The various parameters for a MS/MS spectrum result include spectrum_id, pkl name, peptide sequence, expect value, mh, gi, accession, peptide start, peptide stop, protein definition, modifications, charge state, theoretical mass, p-value and rank.
The .dat file from Mascot (Figure 19) is a text file encoded in MIME format. The file contains the spectrum information along with the search results including fragment information, peptide and protein details. Additionally, it also includes the UniMod XML file which contains average mass and monoisotopic mass pertaining to all the elements, amino acids and post-translational modifications. Figure 18 shows a screenshot of the UniMod XML file. The various parameters for a MS/MS spectrum result from Mascot include spectrum id, hit number, observed mass, mr expt mass, mr calc mass, delta, missed cleavages, ion score, expect value, rank, peptide sequence, gi, protein name, modifications, charge state and pkI name.
Figure 18: A screenshot of the UniMod XML file. The file contains average mass and monoisotopic mass pertaining to all the elements, amino acids and post-translational modifications.

Figure 19: A screenshot of .dat output file from Mascot. The file contains peptide, protein and spectrum information for a given search along with the search parameters.
Parsers were developed in an object oriented fashion in Python programming language. To make the XML parsing fast and efficient, cElementTree module was used for development of XML parsers for X! Tandem and OMSSA. Generic parsers were written for parsing .dat files from Mascot. All these parsers can be imported as a factory and will return the output in the form of objects for further processing. These parsers can be used by any third-party software programs.

4.1.2 Development of data storage models

To enable data processing for combining mass spectrometry data, it was necessary to develop a scalable and extensible, yet efficient data storage models for storing the search results of the mass spectrometry derived data. In this regard, we developed both XML schema and MySQL database for data storage models.

4.1.1.1 XML Schema

A generalized XML schema was developed to store search results from three different search engines including X! Tandem, OMSSA and Mascot. The schema included various fields that include both search engine specific fields as well as general fields. Figures 20, 21 and 22 describes a screenshot of the XML Schema of X! Tandem, OMSSA and Mascot search results.
Figure 20: A screenshot of the XML file containing results from X! Tandem.

Figure 21: A screenshot of the XML file containing results from OMSSA.
Figure 22: A screenshot of the XML file containing results from Mascot.

4.1.1.2 MySQL database architecture

A MySQL database (Version: 5.0.37-community-nt MySQL Community Edition) was created and various tables were created for storing the search results. Tables were created specifically for each search algorithm. Figure 23 describes the schema of the sql tables in detail.
4.1.3 Development of software module for calculating False Discovery Rate

Given the high-throughput nature of mass spectrometry experiments, it is necessary to device methods for validation of tandem mass spectrometry derived proteomic data. The important journals in the field of proteomics including Molecular & Cellular Proteomics (MCP), Journal of Proteome Research (JPR) and Proteomics has already laid the guidelines for publishing data derived from mass spectrometry experiments. The guidelines states that, irrespective of the search algorithm we use, it is good scientific practice to validate the search results. There is a growing concern that some of the results in the literature, particularly from large scale searches, are not reliable. Due to this reason, there is a greater stringency in the reporting of proteomics results.
For large scale experiments, provide the results of any additional statistical analyses that indicate or establish a measure of identification certainty, or allow a determination of the false-positive rate, e.g., the results of randomized database searches or other computational approaches.

Figure 24: A screenshot of the MCP guidelines. The highlighted part shows that, for large scale experiments it is necessary to have additional statistical analyses that indicate a measure of identification certainty.

Proteomics journals have already laid out the stringency in search results as given in their guidelines. MCP was the first to force this after the 2005 workshop in Paris. JPR has already adopted the MCP guidelines. MIAPE – Minimum Information About a Proteomics Experiment and Proteomics journal guidelines are also similar to MCP guidelines. Figure 24 describes a screenshot of the MCP guidelines highlighting the need for statistical validation of mass spectrometry data. The validation methods are
listed in Box 2. The gold-standard for validating this data is the manual inspection of the MS/MS spectra.

The limitations of manual validation are:

- Score distribution depends on multiple factors
  - Instrument, data quality and size of the database
- Subjective
- Needs expertise
- Time consuming and laborious

Using target-decoy database search strategy can be used to calculate False Discovery Rate which is a accepted method of statistical validation of mass spectrometry data. Initially described by Steve Gygi (Elias and Gygi 2007), here the idea is to repeat the search using identical search parameters, against a reversed database. Ideally, we do not expect any significant matches from the reverse database search. So, the number of matches that are found is a good estimate of the number of false positives in the results from the target database. This method is an excellent validation method for MS/MS searches of large datasets. It is not as useful for a search of a small number of spectra, because the numbers are too small to give an accurate estimate of the false discovery rate.

**Box 3: About `esrever` Database**
- Provides a direct estimate of false discovery rate (FDR)
- Requires large dataset to get an accurate estimate of FDR
- Should look like “real” proteins
- Do not contain any genuine matches
- Suitable for MS/MS with enzyme
  - Not suitable for MS/MS without enzyme
  - Genuine y-series to false b-series and vice versa
We want the reverse database to have entries that look like ‘real’ proteins to the search algorithm. However, we want database entries that don’t contain genuine peptide sequences. Box 3 describes the summary of reverse database.

In order to calculate, the MS/MS data has to be searched against the forward and the reverse database. A cumulative distribution of the score/e-value will be created by calculating the number of hits on the forward and reverse database searches. Using these values, the FDR can be calculated as given in Box 4. Software module for calculating FDR for X! Tandem, OMSSA and Mascot has been developed using Python programming language.

**Box 4: The mathematical formula for calculation of False Discovery Rate using search results from forward and reverse database**

For any given score/e-value:

\[
\text{False Discovery Rate (\%)} = \left( \frac{N_r}{N_f + N_r} \right) \times 100
\]

Where:

- \(N_f\): Number of hits in the target database
- \(N_r\): Number of hits in the reverse database
4.2 Development of genome annotation module

Mass spectrometry derived data has a huge potential to be used for genome annotation and has the ability to find key information which can't be found using genomic experiments. A generic flow chart which involves the various steps in genome annotation was laid out. Figure 25 describes a schematic of the generic flow chart for genome annotation.

![Figure 25: A schematic of the various steps involved in the genome annotation workflow.](image)

4.2.1 Development of software for six-frame translation database

Six-frame translation creates all possible synthetic proteins by translating all the open reading frames (ORF) derived from all six reading frames. A software module was created to perform six-frame translation in C++ environment. A Python program was created which will use this software module for creating custom six-frame translated...
database for any given genome sequence. The program returns the data in FASTA format. Box 5 shows a snapshot of a sample FASTA file. The information encoded in the fasta header includes:

1. A unique number assigned to every ORF in the file
2. The number of ambiguous nucleotides (the number of 'N's) that were encountered within the coding region of this ORF
3. The strand the ORF was found on
4. The genomic coordinate for the first NT that starts this ORF off
5. The number of times this ORF sequence was encountered in this chromosome
6. The peptide sequence obtained from the translation

Box 5: Sample FASTA sequence

>gi|45007007|ref|NP_006178.2| 2'-5'oligoadenylate synthetase 3 [Homo sapiens]
MDLYSTPAAALDRFVARRLQPRKEFVEKARRALGALAAALRERGRLGAAAPRVLKTVK GGSSGRGTALKGGCDSELVIFLDCKSYVDQRARRAEILSEMRASLESWWQNPVPGLRL TFEQSVPGALQFRLTSVDLEDWMDVSLVPAMVNLGQAGSGVLDCKSYVDQRARRAEILSEMRASLESWWQNPVPGLRLTFEPQSVPGALQFRLTSVDLEDWMDVSLVPAMVNLGQAGSGVG

4.2.2 Development of a system for submitting searches

Web interfaces were developed using Zope application server and web forms were created using HTML and JavaScript. Figure 10 and 11 describes the web interface for submitting searches to X! Tandem and Mascot. Python programs were created to submit the searches against six-frame translated genome database and protein
database. The next step was to develop a web interface for comparing genome and protein searches. The web interface runs a Python program on the background that compares the two searches and returns the set of peptides uniquely identified in the genome search. Figure 26 describes a screenshot of the web interface for comparing two searches.
4.2.3 Development of modules for genome mapping

Once the peptides are identified through the genome search, the next critical step is mapping the peptide to the genome. This is achieved by using one of the BLAST (Altschul et al 1990) programs - tblastn. Figure 27 describes a schematic of the various BLAST programs and the query & database sequence used. In this regard, BLAST was incorporated into the MS-Analysis suite. The peptide sequence can be mapped to the genome by querying the peptide sequence against the genome sequence using tblastn program. Figure 28 displays a screenshot of the BLAST page in MS-Analysis Suite. Once the peptide is mapped to the genome, the coordinates is used to find whether peptides fall in the
intergenic region using the gene coordinate data of the existing proteins. This would give insights into finding novel genes and splice variants. Additionally, mapping the peptides to the genome will also allow the user to find peptides hitting the intron and un-translated region (UTR) which could give insight into correction of existing gene models and also finding splice variants and alternative translational start sites.

Figure 28: A screenshot of the BLAST page in the MS-Analysis Suite.
4.2.4 Development of GUI for display of search results

In order to analyze the genome and protein searches, a graphic user interface (GUI) was developed to display the search results. This will facilitate the genome annotation workflow. Apart from the GUI for display of search results, a module for creation of custom workbench was created. Multiple workbenches can be created and peptides can be annotated and saved in the workbench. Figure 29 describes a screenshot of the workbench page in MS-Analysis Suite.

![Figure 29: A screenshot of the workbench page in MS-Analysis Suite.](image)

Additionally, a web interface for automated import of xml files from X! Tandem was also created. Figure 30 shows a screenshot of the import page for X! Tandem.

![Figure 30: A screenshot of the web interface for importing XML data from X! Tandem.](image)
4.3 Use of genome annotation module for genome annotation of *Anopheles gambiae* mosquito

The genome annotation module in MS-Analysis Suite was used to perform genome annotation of *Anopheles gambiae* mosquito. This was initiated by the proteomics group at the Dr. Akhilesh Pandey's laboratory at Johns Hopkins University.

4.3.1 Generation of MS/MS data

*A. gambiae* mosquitoes were grown in Johns Hopkins University insectary facility under ambient conditions (humidity 80+5% and temperature 27±1°C). Adult mosquitoes were fed on 10% Karo Dark Corn Syrup and water at least 12 hrs before dissection. Different body parts of adult female mosquitoes such as Malphigian tubules, ovary, midgut and salivary glands were dissected using Olympus SZX12 stereomicroscope. Mosquito tissues were carefully removed in 1X PBS and stored at -80°C until needed. The samples were lysed and run on SDS-PAGE. Bands were excised and digested with trypsin. Mass spectrometric analysis of *Anopheles gambiae* body organs and developmental stages was carried out on LTQ Orbitrap XL high-resolution mass spectrometer.
4.3.2 Designing the annotation workflow

Three different annotation workflows were designed to perform genome annotation of A. gambiae. The results from these workflows put together will give insight into various genome annotation features. Figures 31, 32 and 33 describes the three annotation workflows designed for the genome annotation of A. gambiae.

Figure 31: A schematic of the genome annotation workflow 1.
Protein database search

Peptide list with FDR threshold applied

Peptides mapped on Transcript database

Exonal peptide

Add to the list of exonal peptides

Junctional peptide

Gen ID of the transcript

No hit

1. Junctional peptide from other proteome

2. Junctional peptide from novel gene model

**TABLE 1**

**TABLE 5**

**Figure 32:** A schematic of the genome annotation workflow 2.

Union set of peptides from protein database search and genome database search

Semi tryptic peptides

N terminal semi tryptic peptide

Peptides with acetylated (M) at N terminal

Peptides with (M) as preceding amino acid

Peptides with acetylated (S)/(A) at N terminal

Peptides with any amino acid at N terminal

**TblastN**
(co-ordinates and frame)

C terminal semi tryptic peptide

Peptides with stop codon on C terminal side

Sort out C terminal peptides matching to same gene ID

**Figure 33:** A schematic of the genome annotation workflow 3.
4.3.3 Generation of genome and protein database and database searching

From NCBI ftp site (ftp://ftp.ncbi.nlm.nih.gov/) non-redundant protein data of *A. gambiae, A. aegypti* and *D. melanogaster* was downloaded to create combined protein database. Genome data for genome assembly P3 of *A. gambiae* was downloaded from Vector base (http://www.vectorbase.org/index.php). Six-frame translated genomic sequence was used as a database to search the spectra against genome. Mascot (version 2.2) search engine was used to analyze the data derived from mass spectrometer. Search parameters used are as follows: a) Trypsin as a proteolytic enzyme (allowing up to one missed cleavages); b) Peptide mass error tolerance of 20 ppm; c) Fragment mass error tolerance of 0.1 Da; d) Allowed fixed post translational modification was carbamidomethylation of C. Allowed fixed modifications were oxidation of methionine, acetylation of protein N-terminal,.. The reverse database was used as decoy database to determine false discovery rate. Score corresponding to FDR of 1% was followed as a cut off score.

4.3.4 Execution of the genome annotation workflow using MS-Analysis Suite

All the three genome annotation workflows were executed using the genome annotation module in the MS-Analysis Suite. Peptides obtained after application of 1% FDR cut off were selected for further analysis using computational pipeline developed in our laboratory for identification of novel genes, correction of gene models and validation of known genes of Anopheles. The peptides obtained using
protein database and genome database were pooled for further analysis. tblastn was carried out to obtain genome coordinates. Peptides with more than one hit in genome were separated and were not included in the further analysis using computational pipeline. The peptides without any genome hit, were checked for the exon junction in consent with the predicted gene model. The peptides with a single hit in genome were checked for overlap with the existent gene model. If the peptides were not found to overlap any gene model, they were analyzed as follows. For a given peptide, upstream and downstream region of genome was checked for peptide. The peptides were grouped if they were in the 5kb region. Such regions were given preference in further analysis over the peptides that could not be grouped. The peptides which overlapped Ensembl gene models were further categorized based upon the region of overlap.

4.4 Development of software for handling proteomic and pathway data

Web based software for handling proteomic and pathway data were created using Zope application server, MySQL relational database and Python programming language on a three-tier architecture.

4.4.1 Development of software for handling proteomic data

A semi-automatic data and author tracking system was needed to efficiently track and proteomic data being published in the scientific literature. To meet this need, a web based software to handle proteomic data for Human Proteinpedia was developed. The
software is built on a three-tier client-server architecture including database, middleware and client-side components. The first tier consists of MySQL, a widely used open source database (http://www.mysql.com). Python – an object oriented programming language, is the server-side middleware that controls access to the database and provides additional application logic (http://www.python.org). This architecture is well suited for implementation in the context of biological databases (Navarro et al 2003). The key features of Python include memory management, portability, exception handling and debugging. The client-side component Zope - an open source web application server - provides the graphical user interface and data processing (http://www.zope.org). The MySQL database connection for Python is provided by MySQLdb module (http://sourceforge.net/projects/mysql-python), a Python DB API-2.0 interface. The advantages of Zope over other application servers are its built-in components including web server, web based interface, relational database integration and scripting language support, Document Template Markup language (DTML) (http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/DTML.stx)

4.4.1.2 Development of semi-automatic data and author tracking system for Human Proteinpedia

A database schema was developed using MySQL relational database. A schema consisting of several tables were created for handling the proteomic data from
multiple search algorithms and multiple experiments. A web portal design was created using HTML and CSS. JavaScript was used for server-side scripting.

Using this software, web interfaces were created using which Human Proteinpedia annotator can easily read through the different issues of all the important journals including Molecular & Cellular Proteomics, Journal of Proteomics, Proteomics, Cell, Science, Genome Biology, Analytical Chemistry, Nature, Nature Methods, Nature Biotechnology, Nature Medicine, Nature Immunology, Nature Cell Biology, Nature Genetics, Nature Reviews Cancer, Nature Reviews Genetics, Nature Reviews Molecular Cell Biology, PNAS, Electrophoresis, Journal of Mass Spectrometry, Proteomics Clinical Applications, Clinical Proteomics and Genome Research. Figure 34 describes the homepage of the data and author tracking system. The homepage contains various options including “Add PubMed”, “Add article not indexed in PubMed”, “Annotate existing articles”, “Edit/Delete Annotated articles”, “Approve articles”, “Approved articles” and “Mail manager.” Figure 35 describes the page containing the list of PubMed identifiers to be annotated for gathering mass spectrometry data. Figure 36 describes the page containing article information a form to annotate various data from a given PubMed identifier relevant to the experiment including article title, journal name, publication date, authors, potential principal investigator (PI), PI's institution, PI's e-mail, human sample source, labeling technique used, whether sample is from in-gel, type of the mass spectrometer used, instrument vendor, ionization method, activation method, search algorithm used and estimated data count.
Figure 34: A screenshot of the homepage of the semi-automatic data and author tracking system for Human Proteinpedia.

<table>
<thead>
<tr>
<th>Published ID</th>
<th>Title</th>
<th>Journal of Alzheimer's disease 2005 Apr</th>
<th>Authors</th>
<th>MS instrument</th>
<th>Indexed By</th>
<th>Indexed On</th>
<th>Annotate</th>
<th>Discard</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 13601808</td>
<td>Quantitative proteomics of cerebrospinal fluid from patients with Alzheimer disease.</td>
<td>Zhang, Jing; Goodlett, Dan; Quinn, Joseph; Fendt, Elaine; Karp, Jeffrey; Zhang, Yong; Pan, Catherine; Li, Eugene; Eng, Jenny; Wang, Qin; Aek I, H.; Horvath, Thomas</td>
<td>None</td>
<td>7/25/2006 15:18:14</td>
<td>Annotate</td>
<td>Discard</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 35: A screenshot of the containing the list of research articles along with Journal name, Publication date and PubMed Identifier in the semi-automatic data and author tracking system for Human Proteinpedia.
Figure 36: A Screenshot of Mass Spectrometry based annotations for the PubMed identifier "15851850". The form contains various fields including Journal Name, Article title, Authors, PI of the article and other fields relevant to the mass spectrometry method used.
4.4.1.2 Development of Python data parsers to process proteomic results from different search engines for Human Proteinpedia

The proteomics community uses a range of mass spectrometry database searching algorithms for interpreting mass spectrometry data. In order to facilitate the addition of data into Human Proteinpedia, software programs were needed to efficiently process the large amount of data being generated. In order to address this issue, Python package containing parsers was created in an object-oriented approach to process the mass spectrometry data search results. The software has the ability to process data derived from multiple search algorithms including Mascot, X! Tandem, SEQUEST, Spectrum Mill and OMSSA. The python programs will process the result files and returns the spectrum, peptide and protein data. Additionally, simple utilities listed below were also developed:

1. Convert .pkl files to .mgf files
2. Convert .dta files to .mgf files
3. Merge multiple .pkl or .mgf files to a single .mgf file
4. Count the number of spectra in a given .pkl or .mgf file
4.4.2 Development of software for handling pathway data

A web based software 'PathBuilder' for handling pathway data was created. PathBuilder is built on a three-tier client-server architecture including database, middleware and client-side components. The first tier consists of MySQL, a widely used open source database (http://www.mysql.com). Python – an object oriented programming language, is the server-side middleware that controls access to the database and provides additional application logic (http://www.python.org). This architecture is well suited for implementation in the context of biological databases (Navarro et al 2003). The key features of Python include memory management, portability, exception handling and debugging. The client-side component Zope - an open source web application server - provides the graphical user interface and data processing (http://www.zope.org). The MySQL database connection for Python is provided by MySQLdb module (http://sourceforge.net/projects/mysql-python), a Python DB API-2.0 interface. The advantages of Zope over other application servers are its built-in components including web server, web based interface, relational database integration and scripting language support, Document Template Markup language (DTML) (http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/DTML.stx). More detailed documentation of the architecture including database schema is available at the PathBuilder project website (http://pathbuilder.sourceforge.net). PathBuilder was developed on Linux x86 and tested on Linux and Windows platforms.
4.4.2.1 Designing the database schema

A database schema was designed based on PSI-MI (Hermjakob et al. 2004) and BioPAX (http://www.biopax.org) standards. The schema allows annotation of various reactions characterized as molecular association, catalysis, activation, inhibition and gene regulation upon stimulation with a specific ligand or activation of its specific receptor. Figure 37 describes the schema of the database. It also explains the various database constraints the relationship between the different tables. The various tables in

![Figure 37: A schematic of the database schema of PathBuilder.](http://pathbuilder.sourceforge.net)
the database include \texttt{pathway\_main}, \texttt{molassociation}, \texttt{enzyme\_catalysis}, \texttt{transport}, \texttt{gene\_regulation}, \texttt{molauthority}, \texttt{phy\_entity\_protein}, \texttt{ptm\_standard} and \texttt{taxonomy}. Additionally, it also contains two other tables including \texttt{users\_main} and \texttt{session} which is used for storing the credentials of the users and the session details.

### 4.4.2.2 Development of the annotation workflow, web portal and import/export of pathway data

In order to efficiently implement PathBuilder, an annotation workflow was necessary. The annotation pipeline in PathBuilder (Figure 38) has four central components including annotation of the data, automatic validation of typographical and logical errors, internal review and finally reviews by Pathway Authorities. The installation of PathBuilder provides an unpopulated functional database with default parameters.

![Figure 38: Annotation Pipeline in PathBuilder. A schematic of the annotation pipeline in PathBuilder describing the central components including annotation of the data, automatic validation of typographical and logical errors, internal review and final review (by 'Pathway Authorities').](image-url)
There are two modes of populating the database in PathBuilder. The first mode involves entering of data through a series of web forms. Alternatively, the data in various formats including PSI-MI (Hermjakob et al 2004) and SBML (Hucka et al 2003) can be imported in an automated fashion through a parser written in Python programming language.

Given the necessity of curation of pathway data, PathBuilder was developed primarily for creation of a pathway resource for which the data was entered manually. There are separate web forms available for different data types that allow the user to add, edit or manage data through a web browser. Validation at the level of data entry is achieved through JavaScript for typographical and logical errors, and, any errors found are reported to the user. The default vocabulary used in PathBuilder is based on Biological Pathways Exchange (BioPAX) and Proteomic Standards Initiative’s Molecular Interaction format (PSI-MI) standards, but can be edited to suit specific requirements by modifying the source code. One of the important features in PathBuilder is the ability to administer data easily through a web browser, which permits the annotation process to be carried out at different locations simultaneously.

PathBuilder also incorporates an automated mode of importing large datasets through its Python parser. Currently, PathBuilder supports the standard community exchange formats such as PSI-MI format. For instance, we have successfully imported physical interaction data sets from HPRD (Prasad et al 2009, Mishra et al 2006), IntAct (Kerrien et al) and DIP (Salwinski et al). This would allow researchers in the
biomedical community to collect and aggregate data from various resources which would facilitate creation of a centralized custom database of their interest.
PathBuilder has built-in modules that permit review of the annotated data. Once the reviewer approves an entry, it is marked as 'reviewed' and the entry is finalized in the database. Any change reported by the reviewer is sent automatically to the respective annotator for further changes and the entry is not finalized. PathBuilder also allows a final review and editing by designated scientists who are experts in a specific pathway called 'Pathway Authorities.' The 'Pathway Authorities' report errors if any or specify additional information about a pathway that can be included.

For each annotated entry, PathBuilder automatically records the creation date along with the details of the annotator. Similarly, when an entry is reviewed, the reviewed date and the reviewer name are automatically documented in the database. If an entry is edited after annotation, the last modified date and annotator name are also automatically stored in the database. This allows a user to track the status of an entry in the database.

Web form to browse and do lookup of all pathways available in the pathway resource was developed. The browse and lookup is performed using Python program which connects to the MySQL database. The various fields to do lookup in PathBuilder include gene symbol, protein name, Entrez Gene ID and PubMed ID. Web interfaces were developed to display pathway data under separate tabs such as Physical Interactions, Enzyme Catalysis, Protein Translocation and Transcriptionally Regulated Genes. Software programs were created using PathBuilder to export pathway data to
PSI-MI, BioPAX and SBML formats, which can be used for visualization through other software programs and analysis.

4.4.2.3 Development of software for automated display of pathway network

Software modules were created in PathBuilder for dynamic generation of network graphs that can be viewed through a web browser. The network graphs are generated by Medusa (Hooper and Bork 2005) or Pajek (Batagelj and Mrvar 1998) programs. Medusa is a Java based open source application for visualizing and manipulating interaction graphs. Using Medusa, the graph is displayed in the web browser as an applet. Pajek is a program for analyzing large networks, which displays the graphs as an image in the browser itself. Figure 39A shows the network graph of IL-1 pathway dynamically generated using Medusa or Pajek. The advantage of these built-in tools is that no client side application needs to be downloaded and installed by the users. PathBuilder also provides pathway data that can be visualized using downloadable software such as Cytoscape (Shannon et al 2003) and Osprey (Breitkreutz et al 2003). The pathway data is provided in BioPAX format, which can be loaded into Cytoscape using the BioPAX plugin. The data is also provided as a custom Osprey network .ocf file, which can be loaded into Osprey. Figure 39C-D shows the network graphs of IL-1 pathway rendered using Cytoscape or Osprey.
Figure 39: Display of pathway data. PathBuilder allows dynamic display of all pathway data using Medusa applet which is integrated into PathBuilder. Additionally, PathBuilder allows export of data into standard formats that helps user to visualize pathway data in other software such as Pajek, Cytoscape and Osprey.

4.4.2.4 Development of Application Programming Interface

Sharing and exchange of data between various software applications is important for analyzing pathway data. In this context, an application programming interface (API) was developed in PathBuilder. The API communicates with Python programs in the PathBuilder server and returns the data based on the request. Using the API other software or third party applications can retrieve any pathway dataset available in
PathBuilder. The output of a request to API is a BioPAX, PSI-MI or SBML formatted XML file. The client request to API is specified as URL parameters. This will facilitate third party applications to build modules that can aid visualization, analysis and other simulation based studies. Box 6 describes a detailed documentation on how to use the API in PathBuilder.

**Box 6: HOW-TO: Using the Application Programming Interface (API)**

PathBuilder is installed and running in the http://localhost:8080/PathBuilder.
If you have a qualified domain name or an IP, the URL should be changed accordingly.

The API can be accessed through the URL: http://localhost:8080/PathBuilder/api.

**URL Parameters**

1. The primary URL parameter for the API is `requestType`.

2. To download the list of Pathways available:
   - `http://localhost:8080/PathBuilder/api?requestType=pathwayList`
   - This returns the available pathways along with their `pathway_id`

3. To download the pathway data:
   - In **BioPAX 2.0** format:
     `http://localhost:8080/PathBuilder/api?requestType=biopax2_export&pathway_id=PATH_1`
   - In **PSI-MI 2.5** format:
     `http://localhost:8080/PathBuilder/api?requestType=psimi25_export&pathway_id=PATH_1`
   - In **SBML 2.1** format:
     `http://localhost:8080/PathBuilder/api?requestType=sbml2_export&pathway_id=PATH_1`

**4.4.2.4 Development of NetPath, a resource for human signaling pathways, using PathBuilder**

We used PathBuilder to develop NetPath as a resource for human signaling pathways (Mohan et al Submitted). Pathway data were populated manually using the web forms provided in PathBuilder. Once the pathway data is reviewed, PathBuilder was used to dynamically export data into standard exchange formats and tab-delimited format. The
use of PathBuilder for developing NetPath allowed annotation and review by experts in different countries, most of whom had no bioinformatics expertise. As PathBuilder is extensible, DTML files in NetPath were appropriately modified to accommodate features such as graphical display of a protein molecule. Scripts and DTML files were also modified for creating two broad categories of pathways such as cancer and immune signaling Pathways. DTML files in were created to accommodate features such as FAQs and comments for pathway. Figure 40 shows a screenshot of the NetPath homepage.

![Figure 40: A screenshot of the homepage of NetPath. The homepage contains a web-form to query pathway data, statistics of the available pathway data and a brief description of the NetPath resource.](image-url)