Type 1 Diabetes Software Documentation

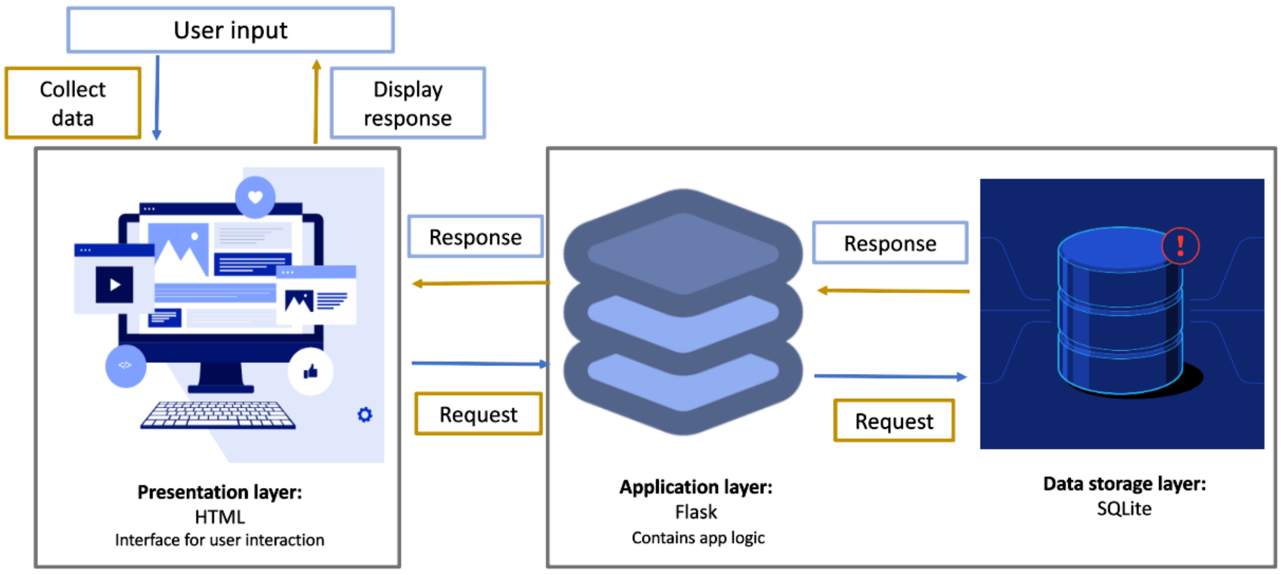
1. Software Functionality

Our software is a web-based application that allows users to search for data related to single nucleotide polymorphisms (SNPs) on chromosome 6 that are associated with increased risk of Type 1 Diabetes (T1D).

Users can search for this information by providing an rsID, a gene name, or a location range on chromosome 6. User input of an rsID returns all available information related to that rsID. Alternatively, input of a gene name will return information for all SNPs associated with that gene. Users can also choose to input a location range for chromosome 6, resulting in the return of information relating to all SNPs located within that range, a Manhattan plot, and an option to choose two or more SNPs for which to calculate downloadable linkage disequilibrium (LD) scores and produce an LD plot.

2. Software Architecture

A three-tier structure is used for the software with a presentation layer, an application layer, and a database layer. The presentation layer consists of the HTML templates that are used to present the user interface to the user. The application layer is built using Flask, which communicates with the SQLite database to retrieve and store user data and processes user requests, generating an appropriate HTML response using the templates. The data storage layer is the SQLite database, which stores all of the information relating to the variants and the gene ontology terms of their mapped genes.



3. Design Principles

3.1. Address the User's Needs

The software must address all of the needs of the user in an efficient and streamlined manner. This is the ultimate objective of the software, and it is therefore critical to have a user-centric approach, keeping in mind all possible outcomes desired by the user throughout the entire software development process.

3.2. Prioritise Useability

Usability of the software was imperative, making a simple, logically ordered application with the minimum amount of decision making necessary for the user to navigate to the desired information. Consequently, navigation forwards and backwards through the software was also important to allow the user to undo and redo actions, aiding overall ease of navigation and access to required data.

3.3. Choose Simple Solutions

Simple solutions were found where possible to address the needs of the user. Simple solutions are better than complex ones as this makes it easier for the software to be understood and implemented by users. Simplicity is particularly important as it is a local software for which the user needs all of the relevant files and must run the application code themselves. This includes the loading of necessary packages, and as such the number of packages used in the software was limited as far as possible.

3.4. Provide All Information in One Place

Reduce the number of pages that the user must access to locate the desired data. Having all of the data returned by a search on a single page reduces the amount of information that user must remember. This is also more user friendly in terms of data extraction, which is the ultimate purpose of the software.

3.5. Handle Errors Informatively

When errors occur due to incorrect usage of the software by the user, provide plain language error messages. Error messages should be polite and precise, giving the user a clear indication of what the problem is and consequently how they can resolve it.

4. Data Sources

4.1. GWAS Catalog

SNPs associated with T1D on chromosome 6 and their relevant information necessary for the software were found in the GWAS catalog. We chose to use the GWAS catalog as it is a comprehensive, standardised, and well-maintained database of genetic associations between various diseases and genetic variants, hence it is useful in identifying SNPs that are associated with a particular disease or trait, in this case Type 1 Diabetes. Having searched for Type 1 Diabetes Mellitus, the associations table was sorted by location and the SNPs found on chromosome 6 were located. Data for five different types of information was extracted for each SNP: rsID, risk allele, p-value, mapped genes, and location, which was modified from the format provided by the GWAS catalog to separate the chromosome number and the location base pair number. This data was formatted into a CSV called Chromosome_6_data.csv.

4.2. CADD

CADD, which stands for Combined Annotation Dependent Depletion, is a computational tool used to predict the functional impact of SNPs. It considers a range of features, including evolutionary conservation, functional annotations, and regulatory regions. CADD has the potential to predict the functional impact of SNPs more accurately than other widely used tools such as SIFT and Polyphen. This is due in part to its ability to integrate multiple sources of information into a single score. SIFT, for example, relies primarily on protein sequence conservation, while Polyphen focuses on the structural and functional effects of amino acid substitutions. This makes CADD a more comprehensive tool for predicting the functional impact of SNPs, although it is important to consider that CADD scores are based on computational models and predictions, which may not always accurately reflect the true impact of a variant in the real world. Another advantage of CADD is its ability to predict the impact of both coding and non-coding SNPs. While coding SNPs have traditionally received more attention, recent studies have shown that non-coding SNPs can also have important functional consequences. CADD considers a wide range of non-coding features, including promoter regions, enhancers, and splicing sites, making it a valuable tool for identifying potentially pathogenic SNPs in non-coding regions. Several of the T1D associated SNPs in our data are found in non-coding regions. For this reason combined with the accuracy of the tool in comparison to other SNP functional impact prediction tools, we decided to use CADD to generate scores for functional impact for the SNPs.

The single nucleotide variant lookup tool on the CADD website was used to collect scaled CADD scores for each SNP. Scaled CADD scores, known as PHRED scores, range from 0 to 48, with higher scores indicating a higher likelihood of functional impact. The CADD model GRCh38-v1.6 was used, with two inputs of chromosome number, which was 6 in all cases, and base pair position of the SNP. The PHRED score relating to the risk allele for each SNP was chosen, and these were added to the CSV file Chromosome_6_data.csv.

4.3. Variant clinical relevance

When determining our measure of clinical relevance for each SNP, following research into how this might be ascertained, we decided that categorising the scaled CADD scores obtained for each SNP would be a good measure. It allows for a better understanding of the impact of an SNP, rather than simply classifying SNPs as either "pathogenic" or "benign", which are the categories often used when assessing clinical relevance of an SNP. According to the CADD website, a scaled score of greater than or equal to 10 indicates an SNP is predicted to be among the 10% most deleterious substitutions that can be done to the human genome, while a score of greater than or equal to 20 indicates the 1% most deleterious. Scaled scores of between 0 to 1 are therefore predicted to be far less deleterious. This information can be used to guide clinical decision-making as it provides a standardized and quantitative measure of variant impact.

As such we decided to have four categories for the scores in terms of risk posed by the SNP:

0-1% Low risk

1-10% Moderate risk

10-20% High risk

20% + Very high risk

No SNPs in our data had a scaled CADD score high enough to be categorised as very high risk, however, we decided to include this category as future development of the application to include T1D associated SNPs from all chromosomes may result in SNPs being identified that do classify as very high risk. Having determined that this was the categorisation scale we would utilise, the category for each SNP was added to the Chromosome_6_data.csv file.

4.4. Risk allele frequency

In order to provide the user a useful and representative reference for the frequency of the risk allele associated with each SNP, we decided to choose three populations from the 1000 genome project, for which extensive allele frequency data has been collected, with high, medium, and low prevalence of T1D. By using multiple populations we maximise genetic diversity, and allow the user to identify SNPs that may be specific or more commonly found in certain populations or ethnic groups based on their rate of T1D. This approach also provides evidence for the role of specific SNPs in increasing T1D risk and improves the confidence in their association with the disease. The decision to use three populations was made as including frequencies for further countries would overcomplicate the data for the user unnecessarily, since three populations adequately represents the entire range of population based T1D rates.

To determine which three populations to choose that fit this model, we consulted the International Diabetes Federation IDF Atlas Reports from 2022, which provided a comprehensive view of prevalence of the disease across 7 global regions. In the graph within the report, if can be seen that across all age groups the European region had the most cases of the disease, and the African region had the least, and the Southeast Asia had a medium prevalence. This helped to inform which countries we should then choose from selection of populations found in the 1000 genomes project. The nations we chose were Finland as the high T1D prevalence country, Japan as the medium T1D prevalence country, and Nigeria as the low T1D prevalence country.

The SNP risk alle frequencies for each country were extracted from Ensembl. For each rsID web page, they were located in the population genetics section. 10 of the SNPs in our data do not have values associated with them for risk allele frequency by country for two reasons. Seven SNPs did not have frequencies on Ensembl for the risk allele linked to their impact in causing Type 1 Diabetes. For example, in one instance frequency information for an SNP was available for the alleles C and T, but the risk allele was A. The remaining three did not have a risk allele associated with them in the GWAS catalog, hence we could not determine which allele to extract the frequency for. We decided to include such SNPs in the data despite this lack of risk allele frequency information as we did not want to exclude 10 SNPs. The risk allele frequency values were added to the CSV file Chromosome_6_data.csv.

4.5. Gene Ontology Terms

Gene ontology (GO) terms provide a link between a gene product and a biological process, cellular component, or molecular function. The Ensembl database was used to identify GO terms for the genes associated with each SNP as it is a comprehensive, accurate, and well-maintained source of information about genes found in the Human genome. The Ensembl tool BioMart was used to extract the required GO data as it is a fast and automated way to retrieve a vast array of annotation data for a specific gene. In BioMart, the filters used were ‘Ensembl genes 109’ under database; ‘Human genes (GRCh38.p19)’ under dataset; ‘chromosome/scaffold, 6’ in the ‘Region’ query filter; and ‘Input external references ID list, Gene name’ in the ‘Gene’ query filter, where a list of all of the gene names that had been previously complied for each SNPs was inputted. The attributes chosen to be displayed were ‘GO term name’, ‘GO domain’ and ‘Gene name’. On the results page, all results from the query were obtained in HTML format and duplicates were by pressing the "Unique results only" button. A compressed file link was emailed by selecting the "Compressed web file (notify by email)" option, which contained the complied results. The data was then reformatted in a CSV file named GO_terms.csv to conform to the needs of our software.

Following the data collection of the terms, it was discovered that some genes had no GO terms associated, while others had more than one. For instances where a gene had multiple associated GO terms we decided to choose one for each category. Through further research we determined that GO terms related to the immune system, found in the biological process category, are most commonly linked to T1D as it is a disease characterized by destruction of pancreatic beta cells by the immune system. Consequently, we searched through the biological process GO terms for each gene, selecting an immune system related term if there was one present. In instances where there were multiple present, and the most specific term furthest down in the GO term hierarchy was chosen. The GO ID of the each chosen term was searched for in the European Bioinformatics Institute web-based tool QuickGo. The co-occurring terms table available for each GO term in QuickGo was then used to select the cellular component and molecular function terms associated with the desired gene that are most commonly co-annotated with the chosen biological process term. This resulted in GO terms that were as relevant as possible to T1D for the user where this was achievable. For instances where no immune system related terms were present in the biological process category for a gene, the first term found in the BioMart list for each category was chosen.

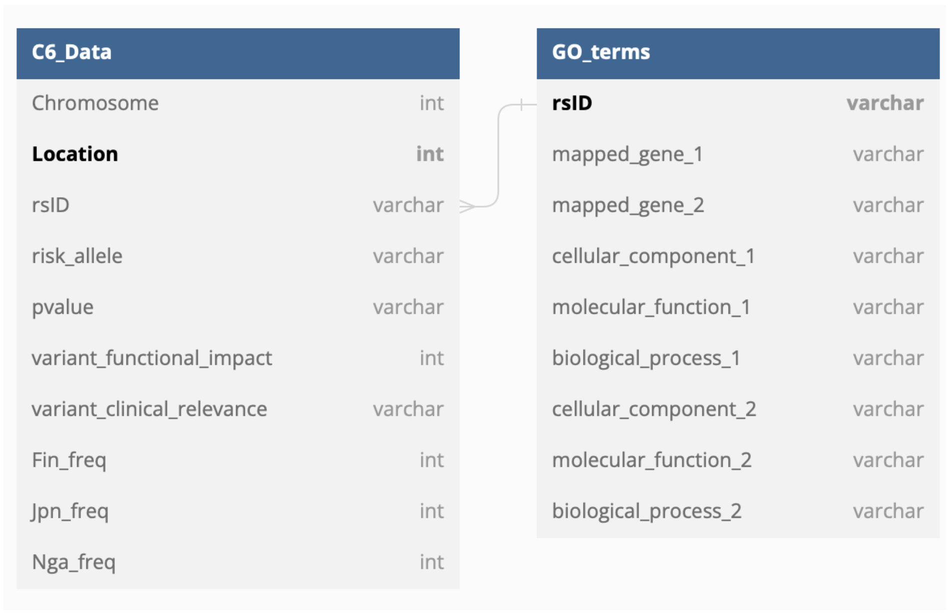
5. Software Technologies

5.1. SQLite

When determining the programming language that would best suit our needs for making a database we decided to use SQL. The reasoning behind this choice is that queries to SQL databases can be run with Python, and we had previously decided to proceed with Python as our programming language for the application. A further decision that had been made was to make our application local and not deploy it online. This combined with the knowledge that our database was would be very small made the self-contained, serverless database engine SQLite the most apt choice for our requirements as it provides sufficient performance for our needs. The code necessary to produce a database using the package SQLite3 is minimal, and the database is stored in a single file that can be queried using Python.

SQLite3 was used to make a single database that contained two tables, one containing all of the data collected as references above relating to each T1D associated variant on chromosome 6, and one containing all of the gene ontology terms associated with each mapped gene. The code written to create the database, makes an empty database file named app_data.db, connects to it, makes two tables named C6_data and GO_terms, and populates these with data read from their corresponding CSV files, named Chromosome_6_data.csv and GO_terms.csv. The primary key for C6_data is location, and the primary key for GO_terms is rsID. These two tables are linked by rsID, with this being the foreign key for C6_data.

The database structure outlined above is expanded upon in the following schema:



C6_Data	
Chromosome	int
Location	**int**
rsID	varchar
risk_allele	varchar
pvalue	varchar
variant_functional_impact	int
variant_clinical_relevance	varchar
Fin_freq	int
Jpn_freq	int
Nga_freq	int

GO_terms	
rsID	**varchar**
mapped_gene_1	varchar
mapped_gene_2	varchar
cellular_component_1	varchar
molecular_function_1	varchar
biological_process_1	varchar
cellular_component_2	varchar
molecular_function_2	varchar
biological_process_2	varchar

5.2. Flask

Flask is a micro web framework which is written in python and it allows full creative control over applications. The flask framework does not require any specific libraries or tools and it does not contain form validation and similar components which would be heavily embedded in other frameworks. Consequently, there can be a plethora of solutions when building a web application and the user is not limited by a specific solution path. However this can come with limitations since the user will need to install specific libraries or import specific modules. Pre-existing libraries which may have been installed in the past, may be older versions and not be up to date, thus it can be easy to damage the OS of the system.

Therefore, as a precaution when working on a flask project it is important to create a virtual environment and begin working on the flask web application. For this specific project a variety of libraries have been imported and the first step was to create a flask instance followed by configuring a secret key. The use of this configuration is to ensure the encryption and protection of sensitive user data. This is important in the instant that we request users to create an account and login in order to use the web application. Therefore, ensuring that the users login details are safely stored in the database.

Following the step of the secret key configuration is implementing the connection between flask and the database built using sqlite3. When creating the connection it is important to ensure that the check same thread value is false. If the database has been created in a different thread then the connection will fail and as such this value is set as false to prevent this error. A cursor is then defined in order to allow query executions and fetching records from the database. Then we started to build the web application by creating flaskform classes each of which was used to retrieve user input and use it as input for the function we created.

Once the class forms were completed routes were formed to bind different URLs to the functions that we created with index being the top level route. The index route was the home page in this application and contained links to other routes where the user is greeted by the input boxes. There were 8 different routes in total, excluding the index route, the first of which requested from the user an rs value or rsID in order to extract information from the database relating to that specific rsID.

This information included the risk allele, p-value, variant functional impact, clinical relevance and other information.

The next route required the user to input a gene name and the function searched the database returning all instances or rows where the specific gene was present. This function returned a table of information similarly to the 4th route which returned information to the user based on searching specific genomic coordinates. For this route the user needed to insert a specific chromosome number followed by a range of base pairs (bps) and the web app returned all genes, rsIDs and other relevant information found within this range of genomic coordinates.

The way this route differs from the rest is that a Manhattan plot was also generated based on the pvalues of the rsIDs which were returned from the genomic coordinate search. Furthermore, the user is also able to calculate the linkage disequilibrium of those rsIDs separately for each population selected. The LD values were calculated and returned into a different route after the user specified which rsIDs to use as the input for calculating the LD. The final part of the code involved 3 routes which were used to download the linkage specificied rsID LD information in the format of a csv file.

5.3. HTML

We decided to use HTML templates to render the user interface because they are easy to customize and allow us to provide a personalized experience to the user. HTML templates are files which may contain static information such as tables, plots or information output in a specific format. However HTML templates can also be used in order to create a more interactive experience as well as make a web application more aesthetically pleasing. However, the most important part is the interactive element that the HTML templates provide. Some of the templates used allow the user to input specific data and as such request specific information. As mentioned in the flask section the user was able to download information based on the LD calculations. The HTML templates allow us to download specific data as csv file instead of downloading everything together. The template language Jinja2 was used in the HTML files to generate HTML input elements and error messages if the user enters invalid input for a field. Aesthetic changes to the software were made using the HTML style attribute to make it more visually user friendly and easier to navigate. Originally, we had planned to use Bootstrap to improve the visual aspect of the software, but the syntax of Jinja2 would not allow for Bootstrap classes to be applied to elements in the HTML.

5.4. Linkage Disequilibrium

Linkage disequilibrium scores are a measure of the correlation between SNPs in a population. A high LD score indicates that two SNPs are highly correlated and tend to be inherited together, and in the case of this software would suggest that the SNP pair is associated with Type 1 Diabetes. LD plots are useful for visually representing the pattern and strength of correlations between pairs of SNPs in a genomic region using R^2 values. The y-axis of the plot denotes the R^2 values and the x-axis denotes the rsID of each user selected SNP. In order to calculate and visually represent linkage disequilibrium scores between SNPs selected by the user in instances where multiple SNPs are returned from a genomic coordinates search, a script for a function named LD was written that, for each of the three populations of interest, Finland, Japan, and Nigeria, would calculate all pair-wise LD scores, output them as a table, and produce a plot of the scores.

The user selected rsID's are provided as input for the function, and a list is subsequently made of all their possible pairwise combinations. We determined that the most efficient way to calculate the scores would be to utilise the online tool LDpop, which is included on the LDlink website, using the website's API. LDpop calculates a number of SNP pair statistics for all populations included in the 1000 Genomes Project, including the R^2 value. The link used for the API call to request the desired information from the LDpop tool includes inputs of the rsID values of the pair, the three populations of interest, FIN, JPT, and YRI, the genome build grch38, and the access token provided by the website upon registration. The output of the API request is wrangled to extract the three R^2 values, one for each population, in a list and multiply them by 100. The R^2 list for each rsID pair is then combined in a single list of lists and subsequently divided into three separate lists containing R^2 values pertaining to each population. A symmetric matrix is made for each population using these R^2 values then wrangled into a dataframe, which is one of the desired outputs for the user.

The package ld-plot was used to produce a heatmap plot of LD scores as a decision was made to exclusively use Python, and we found it to be the best functioning and most intuitive Python package available for this purpose. The inputs for the plot function provided by ld-plot are a symmetric LD matrix that uses the R^2 values, which was created in the LD function prior to the use of the ld-plot function, and a list of label names for the plot, which in this case was the user selected rsID's. The heatmap scale is purple at the lowest end of the range and yellow at the highest end. The ld-plot function does not have customisation parameters, meaning that there were limitations in the modifications we could make to the plot to suit our needs. However, some small modifications were made after the function had been run using the functions title and xticks from the matplotlib package to add a left justified title to each plot and increase the font size of the axis labels to improve legibility of the plot for user convenience. We considered whether to alter the heatmap scale, the range of which is based on the highest and lowest R^2 values rather than the traditional 0 to 100 scale. However, as the majority of the R^2 scores are likely to be very low, we decided that the default scale would be more informative for the user as it shows how the R^2 scores are related to each other for the selected SNPs in a clearer way. With a 0 to 100 scale, in most cases the squares would be purple, and it would therefore be difficult to infer any useful information about how the SNPs relate to each other in terms of LD. However, a consequence of this is that if only two rsID's are submitted by the user, the LD plot will contain a single square that will always be blue, the colour in the middle of the heatmap scale range.

5.5. Manhattan Plot

Script was written within in the Location function to utilise out database and generate a Manhattan plot of the negative logarithm of p-values for all T1D associated SNPs located within a user specified genomic coordinate range. The plot helps to identify significant genomic regions and potential candidate genes that may be involved in the trait of interest.

It constructs an SQL query to select the relevant data for the SNPs located in the input chromosome and base pair range. The query is executed using the pd.read_sql method, which returns a pandas dataframe containing the filtered data. It checks whether the dataframe is empty, and if there is no data within the specified range, it prints an error message and returns without generating the Manhattan plot. The pvalue column of the dataframe is converted to decimal values, as in some cases these values are expressed in scientific notation. The data is then grouped by chromosome and the negative logarithm of the p-value is calculated for each SNP. The chromosome labels, base pair positions, and -log10(p-value) values are stored in three separate lists (labels, x, and y). If there is more than one SNP present in the filtered data, the function uses the matplotlib package to generate a Manhattan plot of base pair position (x) versus -log10(p-value) (y) and labels each point with the corresponding SNP identifier (labels). Subsequently, the plot title and axis labels are also set. If there is only one SNP present, the function prints an error message indicating that no plot is generated.

6. Limitations and Future Development

We found that during data collection there was a lack of information for some of the T1D associated SNPs. Most notably, for ten SNPs information could not be found on Ensembl for risk allele frequency in the three chosen human populations of interest. In addition, some of the genes that mapped to the SNPs did not have GO terms associated with them. In instances where there were GO terms, they were not relevant to T1D in many cases, although we tried to mediate this where possible, and therefore would not necessarily be useful to the user. To enhance the application, further research for sources of the missing data could be carried out to ensure that the user is provided with as much relevant information on their search as possible.

Although our design principles stated that the number of pages the user must navigate through should be reduced as far as possible, we were limited in our ability to align with this principle for the LD scores and plot. We resorted to including a link below the Manhattan plot on the genomic coordinates search results web page that takes the user to a new page with a search box for space delimited rsID input. These rsID's are used to calclate LD scores and output relevant tables and plots on a new web page. Unfortunately, path to reach the desired LD output of the user is more complicated than we had expected it to be, and also requires the user to note down rsIDs from the genomic coordinates search results in order to input them to the LD search box. With further development, the LD search box would be added below the Manhattan plot on the genomic coordinates search results page, so that the user can input the rsID values as seen in the table above and will only need to navigate to one further page to see the results of the LD score and plot computation.

The final design principle for our software was to handle errors caused by incorrect use input appropriately in an informative manner. Although efforts were made to include error handling where possible, due to time constraints we were not able to handle all errors caused by search inputs. Future development of the application would include enhanced error handling capabilities, for example, if a user inputs a word into a genomic coordinates search box, rather than the halting application and error would be returned that allows the user to return to the previous page and correct their mistake.

A simple aesthetic issue that could be solved given further development of the software would be to remove the errant "rs []" that surround each country's LD table and LD plot pair. Given time constraints we were unable to solve this issue, however it does not affect the functionality of the application in any way and is only a cosmetic issue.

Due to the use of the LDlink API in the calculation of LD scores, this can lead to the results of the LD user search taking a long time to appear if a large number of rsID's are searched for. This is a limitation of the software that we decided would have to be an accepted part of the software in its current developmental stage. However, future improvement of the application by software developers could include research into a solution for this problem.

Some aesthetic changes were made to the application to make it more user friendly in line with our design principles, however it does not have a finished, professional look. Given that the software is a functioning prototype, this is an aspect that could be improved further by web designers in the next developmental stage.

We chose not to extend the software to include data relating to T1D associated SNPs on all chromosomes given the timeframe of the project, therefore it is only possible for the user to search for information on SNPs found on chromosome 6. However, given the approach taken to writing the scripts for our software it would be relatively straightforward to extend it in this way in the next stage of development. Data collection for T1D associated SNPs on all chromosomes would be necessary and would be the most intensive element of the software alteration. Subsequently, the database would need to be remade to include the new data, although the script for doing this would remain the same, and the addition of the ability of the location function within the Flask code to search for SNPs in the user input location range on the correct chromosome based on the chromosome number input from the user would be required.

The software is designed to run locally on a computer, meaning that the user must have the necessary files and data downloaded on their machine to in order to use it, all of which are available on GitHub. Following further development by developers and web designers, the final stage would be to deploy the software to the internet, allowing anyone with an internet connection to access it through a web browser. This would be achieved by hosting the software on a web server that is connected to the internet. It is important to note that deployment would require additional considerations both for security and performance.