Course Project Report

1. Introduction

The goal of this project is to create a unified database that brings together gene and genetic variation data from three major genomic platforms: Ensembl, NCBI, and the UCSC Genome Browser. These platforms are essential for genomic research, but they're often used separately, requiring researchers to jump between them to find and compare information. This project aims to solve that problem by building a centralized system that makes accessing gene-related data more efficient and less redundant.

The database will focus on human genes, particularly those associated with diseases. It will compile critical information such as chromosomal locations, exon details, and gene descriptions, giving users the ability to query specific genes and get detailed, organized results. The system will also be designed for future growth, allowing for more genes, genetic variations, and additional attributes to be added over time.

With this system, users will be able to:

- Access detailed information about specific genes, including annotations, chromosomal positions, and exon structures.
- Compare data from Ensembl, NCBI, and UCSC Genome Browser to spot patterns or discrepancies.
- Export search results in formats like PDF or text files for easy analysis.

The project addresses the issue of fragmented genomic data. Currently, researchers have to work across multiple platforms to get a full picture of a gene, which can be time-consuming and error-prone. By integrating these resources into a single relational database, the project aims to provide a streamlined, reliable tool that improves research efficiency.

Designed to be scalable and user-friendly, this system will be a valuable resource for researchers, clinicians, and educators working in genomics, helping them access consistent, comprehensive data and make more informed decisions in their work.

2. Related Work

Integrating genomic data has long been a priority in bioinformatics, with several platforms tackling this challenge in different ways. While these tools are invaluable, they each have limitations that this project seeks to address. Here's a breakdown of existing resources and how they compare to the system being developed.

UCSC Genome Browser

The UCSC Genome Browser is a robust tool for visualizing genomic data, including genes, transcripts, and regulatory elements. Its data tracks and cross-assembly exploration make it a go-to for researchers. However, it's primarily designed for visualization rather than seamless data

integration or export across multiple platforms. As a result, users often need to manually cross-reference its data with other tools to gain a comprehensive understanding of a gene.

NCBI Entrez Gene Database
NCBI's Entrez Gene database excels at providing detailed gene annotations, descriptions, and connections to relevant publications. Its intuitive search interface and programmatic access via a robust API make it highly accessible. However, as a standalone resource, it doesn't integrate with platforms like UCSC or Ensembl. This lack of interconnectivity often forces researchers to handle data alignment on their own.

Ensembl Genome Browser
The Ensembl Genome Browser specializes in delivering precise genomic annotations, including gene locations, transcripts, and variations. Its RESTful API allows for highly specific data retrieval. However, like NCBI, it functions independently and doesn't naturally link to other major genomic databases. This can complicate the task of comparing data across multiple platforms.

Comparison to This Project
This project takes the strengths of existing genomic tools and builds on them to address their limitations. By integrating data from Ensembl, NCBI, and the UCSC Genome Browser into a unified relational database, it offers a more comprehensive and practical solution for genomic research.

One of the key features of this system is its ability to combine data from multiple sources. Unlike standalone platforms like NCBI, Ensembl, or UCSC, this database brings everything together, giving researchers a complete view of gene-related information in one place. It also allows users to create custom queries, making it easier to conduct detailed searches and retrieve specific information.

Another benefit is the system's support for exporting data in easy-to-use format like PDF. This makes it simple to analyze results further or share findings with collaborators. By consolidating multiple sources into a single, user-friendly platform, the database saves researchers time and effort, helping them focus on their work rather than navigating between tools.

3. Requirements

The development of this unified genomic database system was driven by clear goals and specific requirements designed to meet the needs of researchers effectively. These requirements were distilled from the broader project objectives and translated into practical tasks and system functionalities.

Functional Requirements

1. **Data Integration**
    - Connect to Ensembl, NCBI, and UCSC Genome Browser through their APIs to retrieve relevant data.

    - Ensure the data is accurate and consistent by validating inputs and handling errors during integration.
2. **Gene Query Capability**
    - Allow users to search the database by gene_symbol.
    - Provide detailed results, including attributes such as:
        - Chromosomal location (Ensembl).
        - Gene description (NCBI).
        - Exon information (UCSC).
3. **Output Generation**
    - Support exporting query results in a well-organized PDF format.
    - Ensure the exported data is easy to read and includes all relevant details retrieved from the database.
4. **Data Storage and Scalability**
    - Use a relational database structure to store gene-related data efficiently.
    - Design the schema to allow for future expansions, such as adding new data sources or attributes.
5. **Error Handling and Input Validation**
    - Validate user inputs to accept only proper gene_symbol formats (e.g., alphanumeric strings).
    - Handle missing or incomplete data from APIs by displaying clear, informative error messages.

Non-Functional Requirements

1. **Performance**
    - Ensure fast query response times, even as the database scales.
    - Optimize API calls to minimize delays in data retrieval.
2. **Ease of Use**
    - Create an interface that's intuitive and accessible, even for users with minimal technical expertise.
3. **Reliability**
    - Implement robust error handling for potential issues like API failures, invalid inputs, or database connectivity problems.
4. **Maintainability**
    - Use modular code design to simplify future updates, enhancements, and debugging.

System Goals

The primary goal of this system is to provide researchers with a centralized, user-friendly platform for accessing and analyzing genomic data. By integrating information from Ensembl, NCBI, and the UCSC Genome Browser, the system eliminates the need for manual cross-referencing between platforms, saving time and reducing errors. Additionally, it enables comprehensive analysis by consolidating data into one location and offering detailed, customizable outputs in accessible formats such as PDFs. This streamlined approach enhances

research efficiency and ensures that users can focus on their studies rather than navigating multiple data sources.

4. Logical Database Design

The logical database design for this project focuses on organizing data from Ensembl, NCBI, and UCSC Genome Browser into a cohesive relational model. This model defines the structure and relationships of the data, ensuring that information can be accessed and integrated efficiently.

Entity Overview

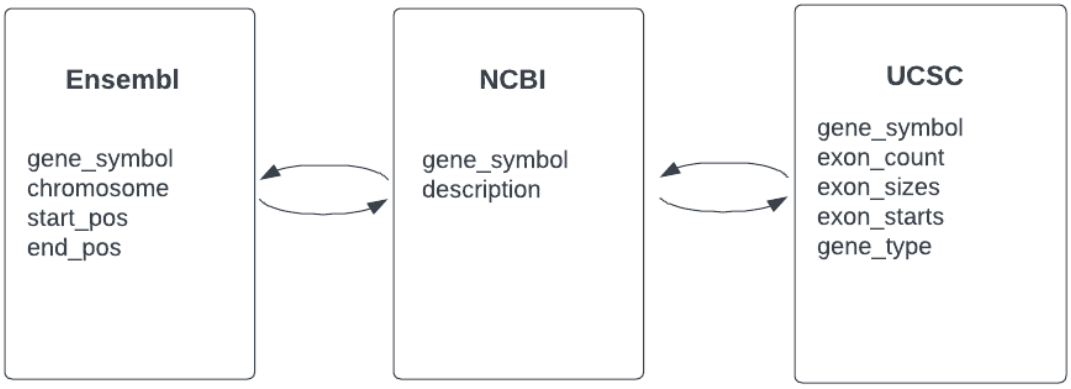
The logical design includes three primary entities, corresponding to the data sources:

1. **Ensembl**: Stores chromosomal information for genes.
2. **NCBI**: Contains gene descriptions and general information.
3. **UCSC**: Includes detailed exon information and gene types.

Each entity is linked by a common key, gene_symbol, which ensures seamless integration and allows for efficient querying across the datasets.

Attributes of Each Entity

1. **Ensembl Table**:
    - gene_symbol: Unique identifier for the gene (Primary Key).
    - chromosome: The chromosome where the gene is located.
    - start_position: The starting position of the gene on the chromosome.
    - end_position: The ending position of the gene on the chromosome.
2. **NCBI Table**:
    - gene_symbol: Unique identifier for the gene (Primary Key).
    - description: A brief description of the gene.
3. **UCSC Table**:
    - gene_symbol: Unique identifier for the gene (Primary Key).
    - exon_count: The number of exons in the gene.
    - exon_sizes: The sizes of each exon, represented as a comma-separated list.
    - exon_starts: The start positions of each exon, relative to the gene start, represented as a comma-separated list.
    - gene_type: The type of gene (e.g., coding or non-coding).



Relationships

- **One-to-One Relationship**:
  - Each gene_symbol is unique across all three entities.
  - This ensures that data for a specific gene can be accurately linked across Ensembl, NCBI, and UCSC datasets.

Benefits of Design

- **Integration**: This design integrates data seamlessly from multiple sources, ensuring consistency and accuracy.
- **Scalability**: The schema allows for the addition of new attributes or tables without disrupting the existing structure.
- **Efficiency**: The relational structure enables complex queries and joins, making it easy to retrieve detailed information for any gene.

The logical database design ensures that the system is both robust and flexible, meeting the needs of researchers while providing a foundation for future expansion. Detailed field definitions for each entity can be found in **Appendix A**.

5. Database Type

The database for this project was implemented as a relational database using SQLite. This choice was driven by the need for a structured, scalable, and reliable system capable of integrating data from multiple sources while maintaining consistency and allowing for complex queries.

Why Relational Database?

A relational database was chosen for this project because it is well-suited to the structured nature of genomic data. Information from Ensembl, NCBI, and UCSC, such as chromosome, start position, and gene symbol, fits naturally into the rows and columns of a relational model. Additionally, the relational approach simplifies data integration by linking information from multiple sources through primary keys, like gene symbols, making it easier to query across datasets. The flexibility of SQL querying also supports complex joins and filters, which are crucial for retrieving detailed and comprehensive gene information.

Why SQLite?

SQLite was chosen as the database management system for this project because it is lightweight, easy to set up, and doesn't require a separate server installation, making it a practical choice for the project's scope. It integrates seamlessly with Perl, the scripting language used for data integration and querying. Additionally, SQLite's self-contained nature—storing the entire database in a single file—simplifies deployment and testing. While it may not be ideal for handling large-scale or high-concurrency operations, SQLite is more than sufficient for managing the current dataset and supporting the project's querying needs.

Benefits

The relational database offers several key benefits for this project. It ensures data consistency by storing all information retrieved from APIs in a structured and uniform format. Efficiency is another advantage, as the use of indexes and primary keys, such as gene symbols, enables fast and accurate data retrieval. Additionally, the database is designed for expandability, allowing new attributes or tables to be added with minimal changes to the existing schema, making it adaptable for future needs.

Limitations

While SQLite is suitable for the current project, there are some limitations to consider for future scalability. It may struggle to handle very large datasets or high levels of concurrent access, so transitioning to a more robust database management system like MySQL or PostgreSQL would be a better option for future expansions. Additionally, SQLite's limited support for concurrent writes isn't an issue for this single-user project but could pose challenges in multi-user environments.

6. Physical/Application Design

The physical design of the database is derived directly from the logical model, translating the conceptual schema into concrete database tables. This section explains how the relational database schema was implemented using SQLite and describes the application design that integrates data from Ensembl, NCBI, and UCSC.

Physical Database Design

The physical design consists of three primary tables: ensembl, ncbi, and ucsc, corresponding to the logical entities. Each table was created with attributes and data types optimized for efficient storage and retrieval.

1. **Table: ensembl**
    - **Fields**:
        - gene_symbol (TEXT, Primary Key): Unique identifier for the gene.
        - chromosome (TEXT): Chromosome where the gene is located.
        - start_position (INTEGER): Start position of the gene on the chromosome.
        - end_position (INTEGER): End position of the gene on the chromosome.
2. **Table: ncbi**
    - **Fields**:
        - gene_symbol (TEXT, Primary Key): Unique identifier for the gene.
        - description (TEXT): Brief description of the gene.
3. **Table: ucsc**
    - **Fields**:
        - gene_symbol (TEXT, Primary Key): Unique identifier for the gene.
        - exon_count (INTEGER): Number of exons in the gene.
        - exon_sizes (TEXT): Comma-separated sizes of exons.
        - exon_starts (TEXT): Comma-separated start positions of exons relative to the gene.
        - gene_type (TEXT): Type of gene (e.g., coding or non-coding).

Primary Key Design

The gene_symbol field is used as the primary key in all three tables to uniquely identify genes and establish relationships between the datasets. Indexes were created on the gene_symbol field in all tables to optimize query performance, particularly for joins and lookups.

Application Design

The application design integrates data from external APIs, processes it, and stores it in the SQLite database. The process consists of three main components:

1. **Data Integration**
    - **API Access**:
        - Ensembl, NCBI, and UCSC data are fetched using REST APIs and Entrez Utilities.
        - Perl scripts handle API calls, parse responses (JSON or XML), and extract relevant fields.
    - **Data Cleaning**:
        - Responses are validated for missing or invalid data, and fallback mechanisms are implemented for partial data.
2. **Database Operations**
    - **Insertion**:

        - Retrieved data is inserted into the respective tables using prepared SQL statements to prevent SQL injection and ensure efficiency.
    - **Querying**:
        - SQL joins are used to combine data from the three tables for integrated queries.
3. **Output Generation**
    - Query results are exported in two formats:
        - PDF: Generated using the PDF::API2 Perl module, with formatted and readable output for each queried gene.

The mapping process involved converting the attributes of each logical entity into database fields with appropriate data types:

Logical Entity	Physical Table	Attributes (Logical)	Fields (Physical)	Data Type
Ensembl	ensembl	gene_symbol	gene_symbol	TEXT
		chromosome	chromosome	TEXT
		start_position	start_position	INTEGER
		end_position	end_position	INTEGER
NCBI	ncbi	gene_symbol	gene_symbol	TEXT
		description	description	TEXT
UCSC	ucsc	gene_symbol	gene_symbol	TEXT
		exon_count	exon_count	INTEGER
		exon_sizes	exon_sizes	TEXT
		exon_starts	exon_starts	TEXT
		gene_type	gene_type	TEXT

7. Test Plan

Testing was an integral part of the project to ensure that the database system functions as intended, providing accurate and reliable results while maintaining usability. The test plan

focused on verifying data integration, querying capabilities, error handling, and output generation.

Objectives

The primary objectives of testing were to:

1. Validate data retrieved from external APIs.
2. Ensure that data is correctly inserted into and retrieved from the SQLite database.
3. Verify the accuracy and formatting of outputs (PDF).
4. Test system behavior under various error scenarios, such as invalid inputs or incomplete API responses.

Test Cases

1. **Input Validation**
    - **Objective**: Ensure that only valid gene_symbol inputs are accepted.
    - **Test Steps**:
        - Input a valid gene symbol (ABCG2) and verify that the system retrieves data successfully.

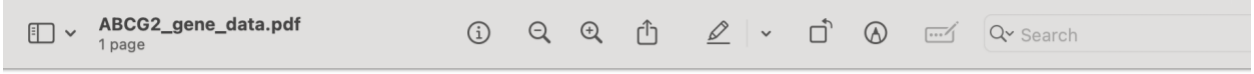
```
vngosb@MacBook-Pro final_project % perl "/Users/vngosb/Desktop/final_project/final_project.pl"
Enter the gene name: ABCG2
Data has been exported to ABCG2_gene_data.pdf.
```

        - Input an invalid gene symbol (123!@) and check for error messages.

```
vngosb@MacBook-Pro final_project % perl "/Users/vngosb/Desktop/final_project/final_project.pl"
Enter the gene name: 123!@
Invalid input. Please provide a valid gene name (example: ABCG2).
```

    - **Expected Results**:
        - Valid input retrieves data without errors.
        - Invalid input generates an appropriate error message without crashing the system.
2. **Database Querying**
    - **Objective**: Confirm that the database correctly integrates data from all three sources.
    - **Test Steps**:
        - Query the gene_symbol of a gene stored in the database.
        - Verify the results include combined data from ensembl, ncbi, and ucsc tables.
    - **Expected Results**:
        - Query results are accurate, complete, and consistent with the inserted data.
3. **Output Validation**
    - **Objective**: Ensure that exported files are accurate and well-formatted.
    - **Test Steps**: Perform a query and export results to a PDF file. Verify that the file includes all relevant details and is formatted correctly.

- **Expected Results**:
  - PDF files are generated successfully and contain accurate data in a readable format.



Gene Symbol: ABCG2
Description: ATP binding cassette subfamily G member 2 (JR blood group)
Chromosome: 4
Start Position: 88090150
End Position: 88231628
Exon Count: 17
Exon Sizes: 2232,83,90,155,125,90,83,251,102,152,158,153,115,60,222,138,635
Exon Starts: 0,4427,5370,7303,9174,11080,17034,23153,24807,27959,31485,40911,41653,42426,
Gene Type: protein_coding

4. **Error Handling**
   - **Objective**: Test the system's response to unexpected errors or failures.
   - **Test Steps**:
     - Simulate invalid database connections and check error messages.
     - Test scenarios where APIs return no results or unexpected formats.
   - **Expected Results**:
     - System handles errors gracefully without crashing, and users receive clear feedback on the issue.

8. Implementation

The implementation phase of this project focused on turning the logical database design into a working system, integrating data from Ensembl, NCBI, and the UCSC Genome Browser, and ensuring users could query the data effectively to get meaningful results. The process involved several key steps, including setting up the database, connecting to external data sources, and making sure the system produced accurate and useful outputs. Throughout the development, there were challenges to overcome, but these experiences also provided valuable lessons that informed the final design. The system was structured using a modular approach, with separate components handling tasks like data retrieval, database interaction, and output generation. This design choice not only made the system more maintainable but also allowed it to scale more easily in the future.

1. **Data Retrieval**:
   - **Ensembl API**: Used to retrieve gene location data such as chromosomal positions, start, and end positions of genes. The data was fetched in JSON format and parsed using Perl's JSON module.
   - **NCBI API**: Used to fetch gene descriptions. Data was retrieved via NCBI’s eutils API and parsed from XML format using Perl’s XML::Simple module.

    - **UCSC API**: Used to fetch exon-related information (e.g., exon count, sizes, and gene type). The UCSC Genome Browser API was accessed via Perl's LWP::UserAgent module for HTTP requests.
2. **Database Interaction**:
    - **SQLite Database**: Data from the APIs was inserted into a SQLite database with three primary tables: ensembl, ncbi, and ucsc. The database was designed to ensure data integrity, with gene_symbol acting as the primary key for all three tables.
    - **SQL Queries**: The system used SQL JOIN operations to retrieve integrated data from multiple tables. A typical query would retrieve gene descriptions from NCBI, chromosomal positions from Ensembl, and exon data from UCSC based on a common gene_symbol.
3. **Output Generation**:
    - **PDF Output**: After querying the database, the system generates a PDF report using the PDF::API2 Perl module. The PDF contains all relevant data in a structured format, making it easy for users to view and analyze the results.

Challenges Faced

1. **Data Inconsistencies**:
    - One of the major challenges encountered was dealing with inconsistencies in the data returned by different APIs. For example, certain genes may be missing data in one database but available in others. To address this, error handling and fallback mechanisms were implemented to ensure that the system still returns the best possible data even when some sources fail to provide information.
2. **API Rate Limits and Reliability**:
    - Both the Ensembl and UCSC APIs had rate limits on the number of requests that could be made in a short time. To mitigate this, the system was designed to handle retries gracefully and to log any API call failures for debugging purposes.
    - Sometimes, the NCBI API returned incomplete or incorrectly formatted data. Parsing these responses required additional validation steps to check for missing fields or unexpected formats.
3. **Database Performance**:
    - While SQLite is lightweight and sufficient for the scope of this project, performance was a concern when querying large datasets. Indexes were created on the gene_symbol field to optimize query performance, but in future work, a more robust RDBMS like MySQL could be considered if the dataset grows significantly.

Lessons Learned

1. **Modular Programming**:
    - One key lesson learned was the importance of modular programming. Separating data retrieval, database handling, and output generation into distinct modules made the development process much more manageable and easier to debug. Each

component could be tested independently, and the system as a whole was more flexible to changes or expansions.

2. **Error Handling**:
    - Robust error handling was critical for ensuring the reliability of the system, especially when dealing with external APIs. Implementing try-catch blocks, logging, and fallback mechanisms ensured that the system could recover from API errors and continue functioning smoothly.
3. **Testing Early and Often**:
    - Testing the system early in the development process allowed us to identify issues related to data format inconsistencies and performance concerns before they became significant problems. Using test cases for each module ensured that the system functioned as expected.

9. Conclusion and Future Work

This project successfully developed a unified database system that combines data from three key genomic resources—Ensembl, NCBI, and UCSC Genome Browser—into a single relational database. The system allows users to query gene-related data and access detailed information, such as gene annotations, chromosomal locations, and exon structures, from multiple sources. By consolidating data retrieval and output generation, the project tackles the problem of fragmented genomic databases and streamlines the research process.

The system pulls data from external sources through API integration, uses SQLite for database management, and relies on Perl for scripting. These components work together to process and store data in an accessible format. Key features like input validation, error handling, and PDF output generation were implemented to ensure users can interact with the system efficiently. The project met its primary objectives: simplifying access to gene data, providing detailed genetic information, and offering a user-friendly interface for researchers.

Future Work

Although the system meets its current goals, there are several areas where future improvements could enhance its capabilities:

1. **Expanding Data Sources**: Future updates could integrate additional genomic databases, such as GENCODE or dbSNP, to expand the range of gene-related information, including genetic variants and mutations.
2. **Adding Genetic Variation Data**: Incorporating data on genetic variations linked to diseases could allow users to explore how specific genetic changes impact gene function or contribute to disorders, enhancing the system's value for genetic research.
3. **Optimizing Performance**: As the database grows, performance may become a concern, particularly with SQLite's limitations. Moving to a more robust relational database, like MySQL or PostgreSQL, would improve scalability and support more complex queries.
4. **Improving Visualization and User Interface**: The current output format PDF is useful, but adding a web-based interface with visualizations could greatly enhance the system’s

usability. Features like interactive gene structure maps or genomic location views would make it more accessible for non-technical users.

5. **Cross-Species Integration**: Future developments could include data from other species, enabling comparative genomics studies. Integrating information from organisms like mice or fruit flies would add valuable context for gene evolution and functional similarities.

This project has successfully addressed the challenge of integrating fragmented genomic data sources into a unified system, making it easier for researchers to query, compare, and analyze gene-related information. The system's modular design ensures that it can be easily extended and updated as new data sources and functionalities are added. While the current version serves as a valuable tool for genomic research, there is plenty of room for future enhancements to broaden its scope, improve performance, and enhance user experience, making it even more beneficial to the scientific community.

References

https://rest.ensembl.org/documentation/info/

https://www.ncbi.nlm.nih.gov/books/NBK25501/

https://www.ncbi.nlm.nih.gov/datasets/docs/v2/api/rest-api/

https://genome.ucsc.edu/goldenPath/help/api.html

https://metacpan.org/pod/PDF::API2

https://www.perlmonks.org/?node_id=808277

Appendix A: Data Dictionary

Table	Field	Description	Source
Ensembl	gene_symbol	Identifier for a gene	Ensembl API
	chromosome	Chromosome where the gene is located	Ensembl API
	start_position	Start position on the chromosome	Ensembl API
	end_position	End position on the chromosome	Ensembl API
NCBI	gene_symbol	Identifier for a gene	NCBI API
	description	Description of the gene	NCBI API
UCSC	gene_symbol	Identifier for a gene	UCSC API
	exon_count	Number of exons in the gene	UCSC API
	exon_sizes	Sizes of exons in base pairs	UCSC API
	exon_starts	Start positions of exons relative to gene	UCSC API
	gene_type	Type of the gene (e.g., coding/non-coding)	UCSC API