Project: **RNA Sequence Analyser**

By Sangini Biswas

Contents:

[**Project Proposal 1**](#_kq9573vza7qb)

[**User Identification 3**](#_2w9z75cwbfkf)

[**Interview 3**](#_7ija21qsm7lh)

[**Objectives and Scope 7**](#_4bkfxooayi48)

[1. User Interface (Web) 7](#_nj9jnn7z3uwe)

[2. Backend for frontend (BFF): 7](#_j1d3yk3w9kyi)

[3. Database: 9](#_x8vvov5ibhlh)

[4. RNA analysis API Layer: 9](#_7px9621rq4eo)

[**Studies & Research 10**](#_dcj43hgr2eed)

[Background 10](#_bb7l5lh3eng1)

[Chemical Composition of the Nitrogenous bases: 10](#_eafnctob6hx)

[Nussinov Algorithm 11](#_611sg8425pzo)

[Comparison 12](#_92wzf20vsxx)

[Similar Programs 14](#_u2exin1h2n2s)

[Nucleic Acid Converter by Stephen Kaminsky: 14](#_2yynewuiiav)

[Protein Translator by Attotron Biosensor Corporation, and Biomodel.UAH.es website - Angel Herráez: 15](#_33bq5crkaa2m)

[RNA Composer by Institute of Bioorganic Chemistry, Polish Academy of Sciences, Institute of Computing Science - Poznan University of Technology: 16](#_7ckka1r1zf4i)

[QR Codes for each website respectively: Nucleic Acid Converter, Protein Translator, and RNA Composer 17](#_ou8ncmd0lqay)

[Application Stack Research 18](#_s87szce1rdia)

[Solution and Why? 18](#_7my79v3ssd0s)

[**Application Stack 20**](#_1icypefm4rdo)

[**Database Installation and Structure 20**](#_jwdhrtxkfgfv)

[Entity-Relationship Diagram (ERD) 21](#_gzwl6zn5t5ri)

[Data Definition Language (DDL) 21](#_2e8m7d7rufva)

# Project Proposal

This project aims to design and develop a user-friendly RNA codon sequence analysis application. By studying existing tools and their features, this application brings most of the elements in a single place by providing them as decoupled and modular as possible.

Explaining the complexities by understanding the codons becomes critical knowledge in producing information about gene expression, health and disease. This has been an objective for many researchers, universities and companies who have invested in building software and libraries that help to understand the complexities.

Let’s try to understand in simple terms. Assume DNA is a recipe book, and the recipes within this book are the proteins that our body builds. This translates to the instructions mentioned in the recipe book as the DNA and the ingredients and steps that are the proteins translated as RNA.

Within this RNA, there are 64 different codons, each made up of a three-letter sequence of the following four bases:

* A: Adenine
* U: Uracil (replacing thymine found in DNA)
* C: Cytosine
* G: Guanine

So, the sequences build a language that provides critical information about protein synthesis. Hence, it becomes vital for scientists to examine the order of frequency of these codons within an RNA sequence.

Some examples of what helps in understanding would be:

* **Gene expression:** The rate at which codons are used within a gene influences how much protein is produced. Studying the pattern provides information about cellular processes and potential disease markers.
* **Translation efficiency:** The “codon bias” analysis, i.e. the preference of specific codons, helps predict how protein production affects cell functioning.
* **Evolutionary adaptations:** “Survival of the fittest” or adaptation of organisms by understanding the codon bias provides an understanding of how organisms cope with different conditions.

Within the existing landscape of available software, it provides most of the answers by doing the RNA codon analysis. Examples:

* EMBOSS cusp: A versatile command-line tool offering various codon analysis utilities (codon frequency, bias calculations, etc.) (<https://www.bioinformatics.nl/cgi-bin/emboss/cusp>)
* Geneious Prime: A commercial software featuring a user-friendly interface and advanced codon analysis features (<https://www.geneious.com/download/>)
* Codon Usage Database: A comprehensive online resource providing codon usage data for various organisms (<https://www.kazusa.or.jp/codon/>)

This application is developed to offer:

* A user-friendly interface (UI): This is designed so that students, scientists or researchers with varying levels of coding expertise can use the UI for an easy interaction.
* An abstracted backend: This is designed to keep the authentication, authorisation, data persistence, and profile maintenance decoupled with the UI and analytical application.
* A database: This is designed to contain information about the users, the RNA sequences they are inputting, and the output of their analysis so that the users can refer back for comparison purposes.
* RNA analysis application: This is designed to expose the sequence analysis features as REST APIs that can be accessed by external developers who understand the APIs. They are also integrated with the backend of this application to provide the analysis output to the UI.
* Use Open-source libraries and make this application available over GitHub for feedback and enhancements.

# User Identification

My user is my teacher, Mr McMahon, my A-level biology teacher at CRGS. Mr McMahon had difficulty teaching us how the entire strand of RNA is formed based on a chart that showed different combinations of triplet codons so I aim to create a program to aid his teaching so he can present the content in a visual form which will help many A Level, or GCSE students, as visual representations of what things look like help learners to remember the topic. This is to be both a lesson and revision resource where students can consolidate their learning but Mr McMahon can also use it in his lessons as an aid.

# Interview

1. **Can you provide a brief overview of the project requirements for developing an RNA Sequence Analyzer?**

The project should involve creating an RNA Sequence Analyser using Python for the backend logic, JavaScript for the frontend, and MySQL for data storage. The primary goal is to analyse and predict RNA secondary structures efficiently. The project should support the retrieval, storage, and analysis of RNA sequences and their associated metadata.

1. **How do you plan to approach the development of the RNA Sequence Analyzer? Can you outline the key technical aspects to include?**

It should start by designing a robust MySQL database schema to store RNA sequence data and relevant metadata. For the backend, implement algorithms like Nussinov or Zuker for RNA secondary structure prediction. The frontend will provide a web-based user-friendly interface for querying and visualizing the results.

1. **How do you plan to handle data storage in MySQL and ensure efficient retrieval of RNA sequences for analysis?**

The database should be normalised and proper indexing used to optimise query performance.

1. **The project involves implementing algorithms like Nussinov or Zuker. Can you discuss the algorithm implementation in Python and how to integrate these algorithms into the RNA Sequence Analyzer?**

The selected algorithms will be integrated into the backend of the application. These algorithms will be used for predicting RNA secondary structures, and the results will be stored in the MySQL database for later retrieval and analysis.

1. **How do you envision the user interface for the RNA Sequence Analyzer, and what features will it include to make it user-friendly?**

The user interface will have features for submitting RNA sequences, viewing analysis results, and visualizing predicted secondary structures. It will be designed with simplicity in mind, with clear navigation and interactive visualization tools to enhance the user experience, especially with students being the main user of this project.

1. **Ensuring accuracy is crucial. How do you plan to handle errors and implement quality assurance measures in the development process?**

There should be robust error-handling mechanisms implemented in the Python code to catch exceptions and provide informative error messages. Additionally, make sure there is thorough testing, including unit testing and validation checks, to ensure the accuracy of the implemented algorithms and the overall system.

1. **Security is a priority for us. How do you plan to address security concerns, especially when dealing with user-submitted RNA sequences and sensitive data?**

Security is paramount. Input validation to prevent SQL injection and other security vulnerabilities should be included. Follow the best practices for securing user data and maintaining the confidentiality of the RNA sequences and analysis results.

1. **Can you explain how to handle energy parameters in the calculations and how these will be stored in the MySQL database?**

Energy parameters, such as penalties for unpaired bases or energies for different types of base pairings, will be defined based on the chosen energy model. These parameters will be used in the calculations and stored on a separate table in the MySQL database. This table will provide flexibility to adjust energy parameters without modifying the core code.

1. **The Zuker algorithm is known for energy minimization in RNA secondary structure prediction. How do you plan to integrate Zuker's approach into the application, and what considerations will you take for energy calculations?**

The Zuker algorithm considers energy parameters for various interactions. The energy calculations will be based on the chosen energy model, and the results will be stored alongside the secondary structure information in the MySQL database.

1. **How do you plan to implement the Nussinov algorithm for RNA secondary structure prediction in Python? Can you walk us through the key steps and considerations?**

The Nussinov algorithm involves dynamic programming to find the maximum number of base pairs in a given RNA sequence. Create a Python function that iteratively fills a 2D matrix, considering different cases for base pairings and unpaired bases. The resulting matrix will represent the optimal secondary structure and store this information in the MySQL database for later retrieval.

# Objectives and Scope

After a comprehensive study and research of existing systems and software, it has been observed that an application that can provide a common platform for enhancements, a friendly user interface, and a modular REST API package to integrate with existing bioinformatics libraries is needed as an open-source that is readily available online for community enhancements.

### User Interface (Web)

* 1. As a user, I should be able to launch the application as a web page and browse through the page's content to understand the application.
  2. As a user, I should be able to view the credits towards building this application.
  3. Registration
     1. As a user, I should be able to register by providing a username (email), password and information like role (student, company, university, etc.) and contact details.
     2. As a user, I should be able to verify email post registration by providing a token.
     3. As a user, I should be able to change the password using my registered email for verification.
  4. Login
     1. As a user, I should be able to log into the application by giving my registered username and password.
  5. RNA sequence analysis
     1. As a user, I should be able to view a list of RNA sequences added earlier for analysis. On selecting a sequence, it should provide the analysis data. I should be able to update the sequence and save it.
     2. As a user, I should be able to add new RNA sequence for analysis. When submitted for research, it should show the analytical features data.
     3. As a user, I should be able to visualise the RNA sequence strand.

### Backend for frontend (BFF):

* 1. As a system, it should expose WEB interfaces as REST APIs.
  2. As a system, it should ensure that any calls to REST APIs are provided with valid client API tokens.
  3. As a system, it should provide relevant error messages during exceptions.
  4. Database initialisation
     1. Initialise the database connection
     2. Create the database schema and model it if the schema is not available. This is to ensure that this application can be run without explicitly knowing about configuring the schema manually.
  5. Registration
     1. As a system, it should save the registration data into the database, ensuring the password is encrypted using the password given by the user during registration.
     2. As a system, it should send an email to the user to verify their email by providing a token.
     3. As a system, it should update the password by encrypting it using the new password.
  6. Login
     1. As a system, it should validate credentials using the username and password.
     2. As a system, it should show validation messages due to any invalid credentials provided.
     3. As a system, it should take the user to the logged-in page after successful login.
     4. As a system, it should maintain a login audit for login attempts.
     5. As a system, it should lock the user after 3 login attempts and send an email for password reset.
  7. RNA sequence analysis
     1. As a system, it should verify that the user is in a logged-in state to access the RNA sequence analysis features.
     2. As a system, it should call the RNA analysis APIs to get the bioinformatics data.
     3. As a system, it should save the bioinformatics data for the logged-in user and the RNA sequence submitted to the database.
     4. As a system, it should be able to fetch the data from the database.

### Database:

* 1. Provide with relevant schema to store
     1. User profile
     2. User credentials
     3. User audit trails
     4. User RNA analysis data
  2. Create the required schema for the above requirements.
  3. Create the required roles and privileges for accessing the database.

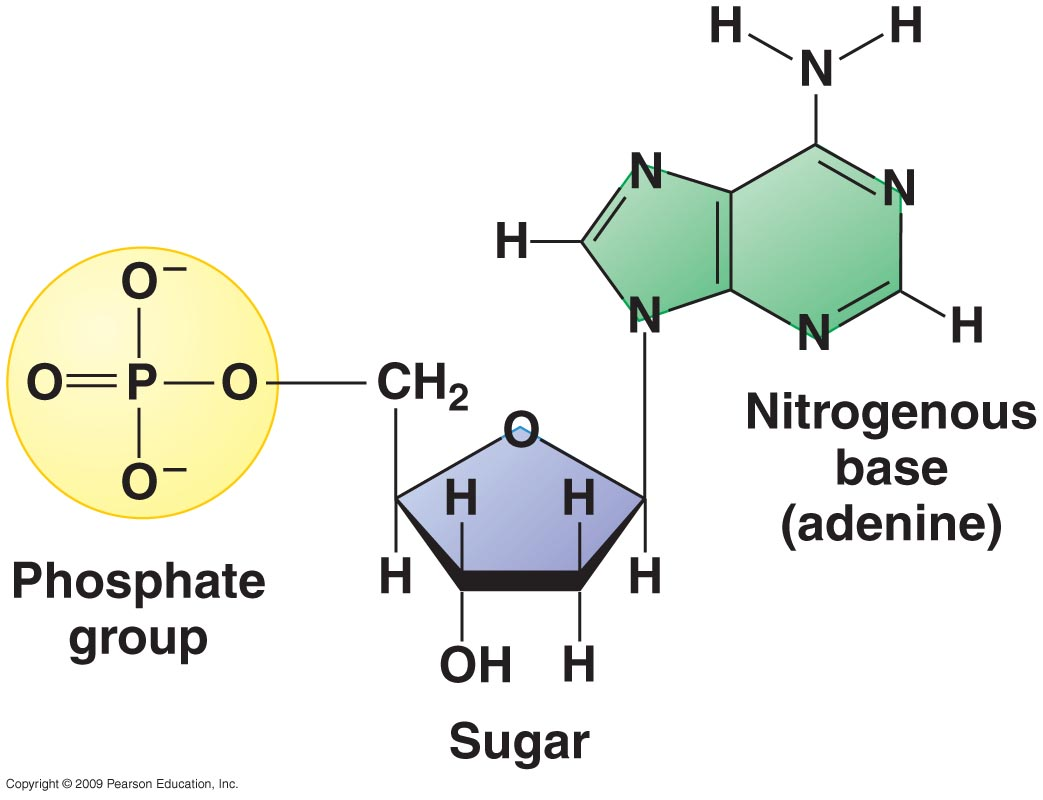
### RNA analysis API Layer:

* 1. As a system, it should ensure that any calls to REST APIs are provided with valid client API tokens.
  2. As a system, it should provide relevant error messages during exceptions.
  3. As a system, it should provide any response in JSON format.
  4. As a system, it should be running in a separate application server so that external applications can use the REST APIs by accessing the APIs directly. This system should ensure that the external systems have been provided with valid client tokens to access these APIs.

# Studies & Research

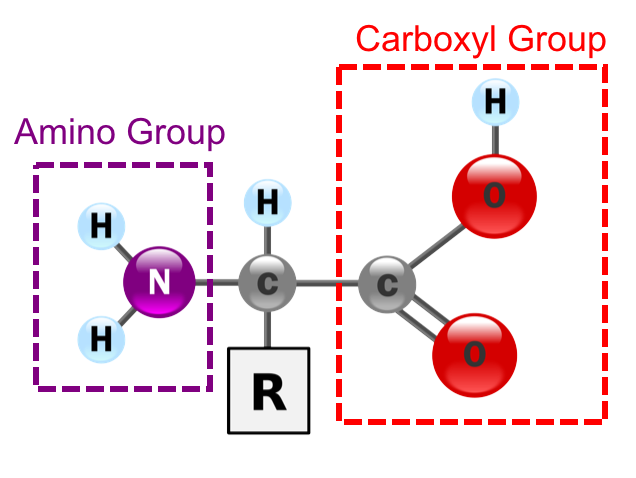
### Background

RNA is made up of smaller units called nucleotides. Each nucleotide consists of three components: a sugar molecule, phosphate group, and a nitrogenous base. There are four types of nitrogenous bases in RNA: adenine (A), cytosine (C), guanine (G), and uracil (U).



RNA helps build proteins by carrying instructions for assembling amino acids. Amino acids join together to create proteins.

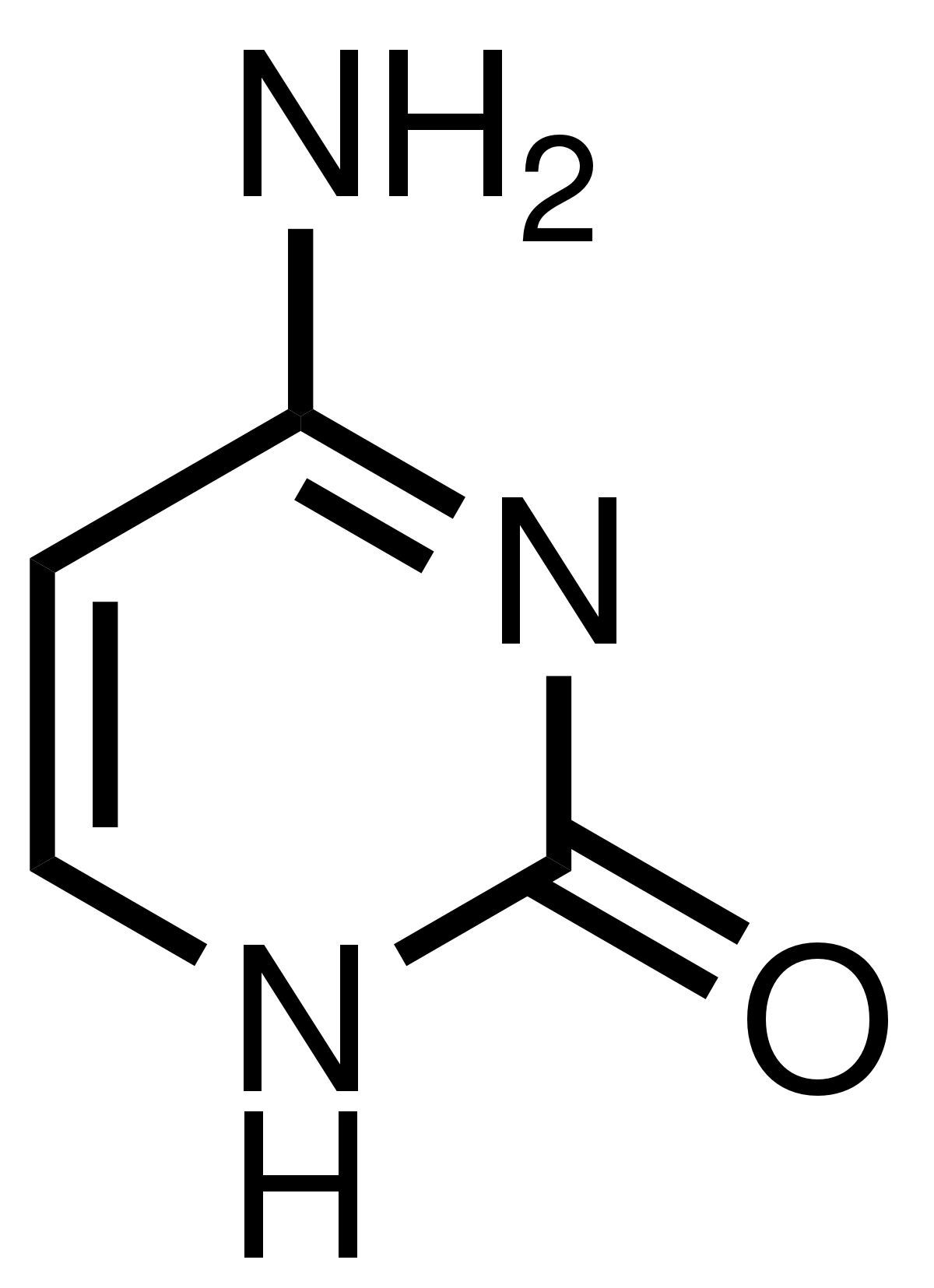
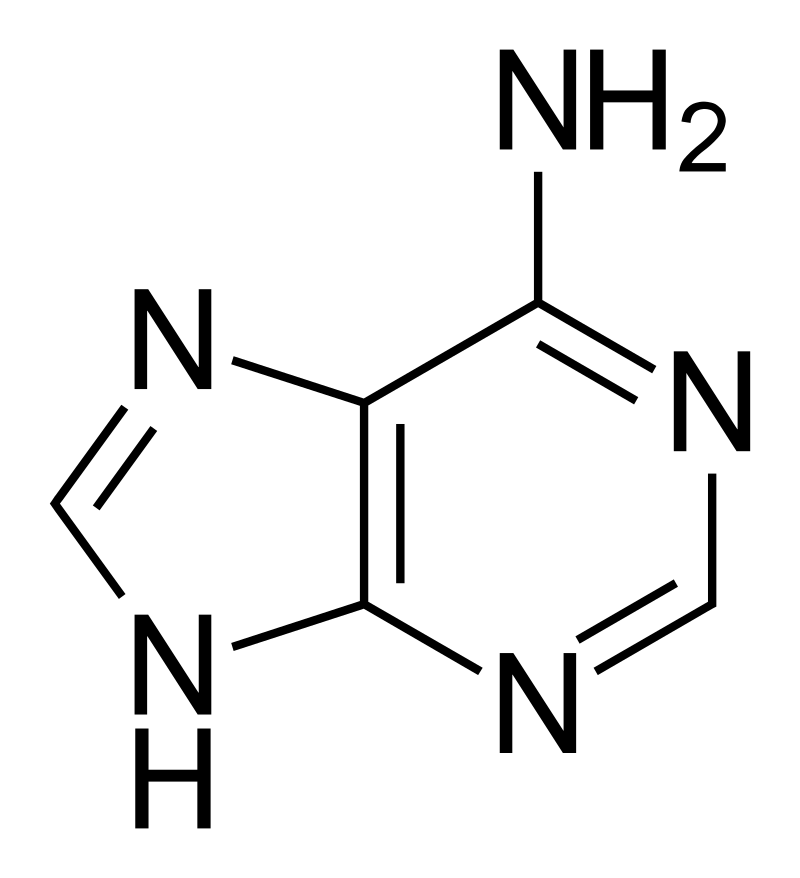
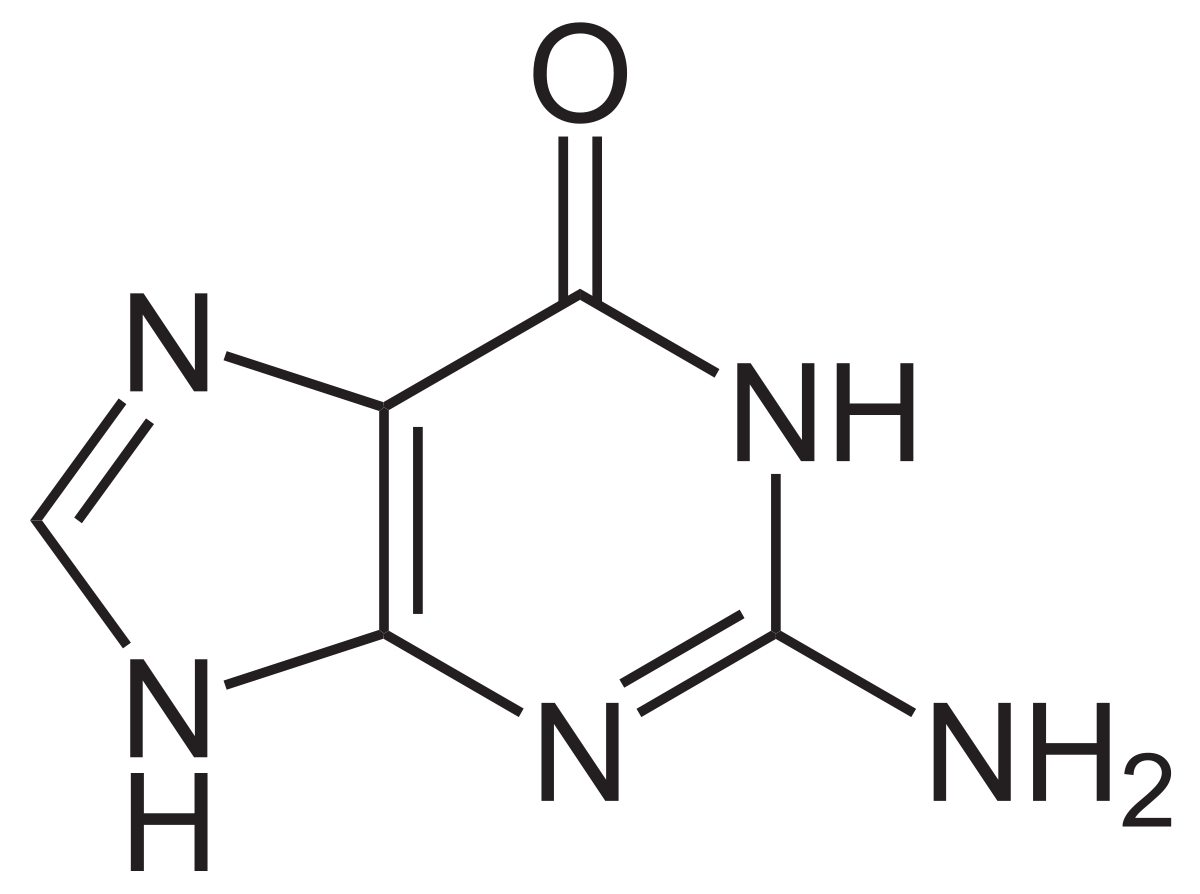
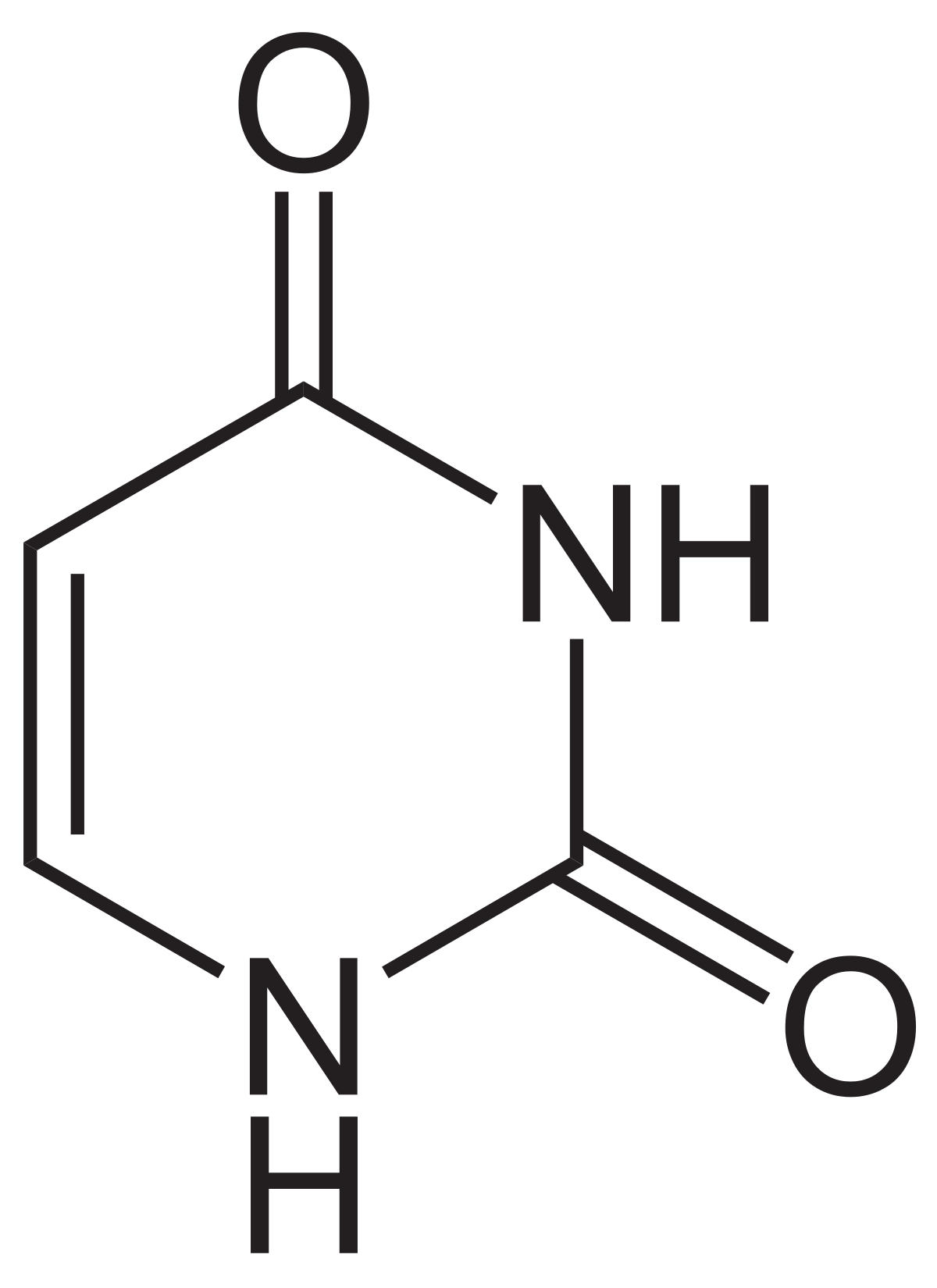
They have a common structure: an amino group (NH2), a carboxyl group (COOH), and a unique side chain or R group. The side chain distinguishes one amino acid from another.



Triplet codons in RNA are sequences of three RNA nucleotides that act as the instructions for assembling amino acids in the correct order to build proteins. The specific sequence of codons in an RNA molecule determines the sequence of amino acids in the resulting protein, allowing cells to create a wide variety of proteins with different functions.

## Chemical Composition of the Nitrogenous bases:

*Adenine: Uracil: Guanine: Cytosine:*



RNA bases pair with one another through hydrogen bonds. Adenine (A) pairs with uracil (U), and cytosine (C) pairs with guanine (G). The pairing between A and U, and C and G follows specific base-pairing rules. These rules ensure that the RNA molecule forms a stable and predictable structure. A always pairs with U, and C always pairs with G.

The specific sequence of complementary base pairs in an RNA molecule leads to the formation of secondary structures, such as hairpin loops and stem-loop structures, bulges, internal loops, and pseudoknots. These structures play crucial roles in RNA functions, including binding to other molecules, regulating gene expression, and catalysing chemical reactions. The number and strength of hydrogen bonds between base pairs affect the stability of an RNA strand. A higher number of hydrogen bonds results in greater stability. For example, a G-C base pair has three hydrogen bonds, while an A-U base pair has two. This stability is important for the overall structure and function of RNA molecules.

## **Nussinov Algorithm**

The Nussinov algorithm is a classic computational method used for predicting the secondary structure of RNA molecules. It is one of the earliest and simplest dynamic programming algorithms for RNA secondary structure prediction.

In simple steps:

1. *The input to the Nussinov algorithm is the primary sequence of an RNA molecule as a sequence of RNA bases (A, U, C, G).*
2. *The algorithm uses a dynamic programming approach to compute the maximum number of base pairs that can form in the RNA secondary structure. It constructs a square matrix of size N x N, where N is the length of the RNA sequence.*
3. *The diagonal and the entries just above the diagonal of the matrix are initialised to zeros. These cells represent the minimum loop length, and they are set to zero because they do not form base pairs.*
4. *The algorithm iteratively fills in the remaining cells of the matrix in a bottom-up manner, considering all possible base pairings. For each cell M[i][j] in the matrix, the algorithm finds the maximum number of base pairs that can form between positions 'i' and 'j' in the RNA sequence and it checks if the RNA bases at positions 'i' and 'j' can form a base pair.*

*If they can, the algorithm considers the option of pairing them and checks the maximum number of base pairs that can form in the region between positions 'i' and 'j.' This is done by examining the cells in the matrix within the range (i+1, j-1) and adding 1 to the count for the base pair formed at positions 'i' and 'j'.*

**Zuker Algorithm**

The Zuker algorithm, also known as the RNA Mfold algorithm, is a method for predicting the secondary structure of RNA molecules. The Zuker algorithm aims to find the secondary structure that minimises the free energy of the RNA molecule.

It works as such:

1. *The input to the Zuker algorithm is the primary sequence of an RNA molecule, represented as a sequence of RNA bases (A, U, C, G).*
2. *The algorithm relies on a set of experimentally derived energy parameters, which quantify the stability of various types of base pairs and structural motifs in RNA. These parameters include stacking energies for adjacent base pairs, loop penalties, and penalties for non-canonical base pairs. These parameters are used to compute the free energy associated with a given secondary structure.*
3. *Similar to the Nussinov algorithm, the Zuker algorithm employs a dynamic programming approach to find the secondary structure with the minimum free energy. It constructs a two-dimensional matrix (usually referred to as 'V') of size N x N, where N is the length of the RNA sequence.*
4. *The algorithm initialises the matrix by filling in the diagonal elements with zeros, representing the minimum energy for unpaired nucleotides.*
5. *The core of the Zuker algorithm involves filling in the remaining cells of the matrix in a manner that minimises the free energy of the secondary structure. This recursive algorithm computes the minimum free energy for each subsequence of the RNA sequence.*

*For each cell V[i][j] in the matrix, where i < j, the algorithm considers all possible base pairs that can form between positions 'i' and 'j' in the RNA sequence. It calculates the energy associated with each base pair, which includes contributions from stacking interactions, loop penalties, and non-canonical pair penalties. It recursively evaluates the energies of the substructures on either side of the considered base pair. It sums these energies with the energy of the base pair itself to compute the total energy for the structure.*

*It then selects the base pair with the lowest total energy for each cell V[i][j], identifying the most stable secondary structure element. The minimum free energy for the entire RNA sequence is found in the top-right corner of the matrix (V[1][N]), representing the entire sequence's secondary structure's stability.*

1. *After filling the matrix, the algorithm performs a traceback to determine the actual secondary structure that minimises the free energy. Starting from V[1][N], it follows a path that selects base pairs and identifies loop regions until it reaches the bottom-left corner of the matrix.*
2. *The final output of the Zuker algorithm is the predicted secondary structure of the RNA molecule, represented in dot-bracket notation, where paired bases are indicated with parentheses (e.g., "(((....)))"). The structure's stability or free energy is also reported.*

## **Comparison**

The Zuker and Nussinov algorithms are both used for predicting the secondary structure of RNA molecules, but they employ different approaches and have distinct characteristics:

* **Programming:** The Zuker algorithm uses a dynamic programming approach to predict RNA secondary structures. It considers the minimum free energy of the secondary structure and employs experimentally derived energy parameters to evaluate the stability of various structural motifs.

The Nussinov algorithm also uses dynamic programming, but it predicts secondary structures based solely on base pairing and sequence complementarity. It focuses on maximising the number of base pairs rather than evaluating free energy.

* **Energy Parameters:** The Zuker algorithm relies on experimentally derived energy parameters to calculate the free energy of RNA secondary structures. It considers stacking interactions, loop penalties, and penalties for non-canonical base pairs. This makes it more accurate for determining the thermodynamic stability of structures.

The Nussinov algorithm does not consider energy parameters. It simply counts the number of base pairs in the predicted structure, making it a simpler and faster algorithm but less accurate in terms of energy considerations.

* **Accuracy:** The Zuker algorithm tends to provide more accurate predictions of RNA secondary structures due to its consideration of thermodynamic stability.

The Nussinov algorithm provides quick and reasonably accurate predictions for basic secondary structures but may not capture the thermodynamic intricacies of RNA folding as well as the Zuker algorithm.

* **Applications:** The Zuker algorithm is computationally more intensive due to the evaluation of energy parameters. It can be slower for large RNA sequences or when complex structures are involved. It is often used for detailed RNA structure analysis, RNA folding kinetics, and predicting the stability of complex RNA structures. It is suitable for research involving the thermodynamics of RNA folding.

The Nussinov algorithm is computationally simpler and faster, making it suitable for large-scale predictions and initial structure assessments. It is commonly used for quick and approximate predictions of RNA secondary structures. It is useful for initial structure assessments and as a building block for more advanced algorithms.

### Similar Programs

#### Nucleic Acid Converter by Stephen Kaminsky:

Advantages:A screenshot of a computer

Description automatically generated

* Easy-to-follow UI
* Easy instructions
* Simple to use for any age audience
* Has shortcut links to skip straight to DNA or mRNA conversion
* Downloadable file so it can be used locally - as it is local there is less privacy concern
* Has a contact page to contact for any issues
* No limit on the amount of letters the user can type
* Easy to enter in bases that will be generated to create, in this case, DNA, mRNA, and protein
* Fulfils the role of a converter where it will take in an input and just convert it as an output

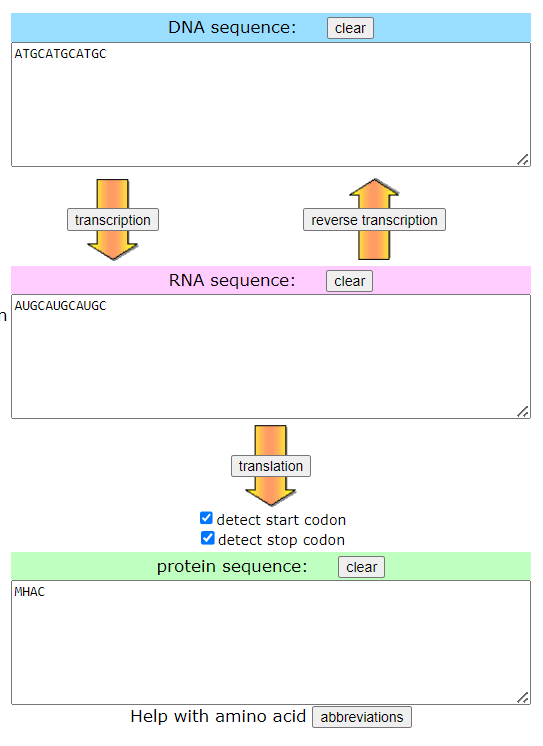
Disadvantages:

* UI design itself is not very appealing - darker colours with the harsh contrast with the blue can affect some viewers especially if they are using this at night and this can strain the eyes
* The website itself is not very accessible - when I was conducting this research this was a difficult website to find and it had not been one of the first search results
* As it is only a converter, it does not visually display the strand created; my project aims to solve this issue by providing in text form, as this project has done, but also a visual creation of what the user has simulated
* Does not provide information on what RNA, DNA or proteins are - making the assumption that students while studying and using this as a revision resource will already know the topic beforehand when it would be more useful to have concise information just to refresh their memory
* No method of saving or accessing previous creations

#### Protein Translator by Attotron Biosensor Corporation, and Biomodel.UAH.es website - Angel Herráez:

A screenshot of a computer

Description automatically generated

Advantages:

* Easy-to-follow UI
* Easy instructions
* Simple to use for any age audience
* No limit on the amount of letters the user can type
* Easy to enter in bases that will be generated to create, in this case, DNA, mRNA, and protein
* Fulfills the role of a converter where it will take in an input and just convert it as an output.
* Has small descriptions of the roles of DNA, RNA and proteins as well as the full forms of DNA and RNA
* Has arrows between all boxes to show how they convert from one another: for DNA -> RNA transcription must occur and for RNA -> protein translation must occur. RNA -> DNA goes through reverse transcription
* Website was easy to find - during research it was on the first page of results

Disadvantages:

* UI design itself is not very appealing - no formatting of text font or font size, and plain colours are used
* As it is only a converter, my project aims to solve this issue by providing in text form, as this project has done, but also a visual creation of what the user has simulated
* No downloadable file to use locally incase the user cannot access the website anymore
* No contact link or email to use if there is an issue with the website
* No method of saving or accessing previous creations

## 

#### RNA Composer by Institute of Bioorganic Chemistry, Polish Academy of Sciences, Institute of Computing Science - Poznan University of Technology:A screenshot of a computer Description automatically generated

A screenshot of a computer

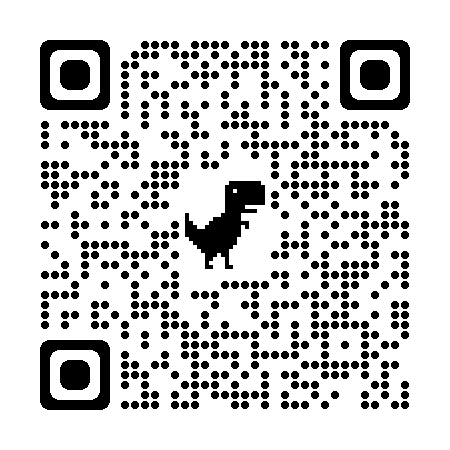
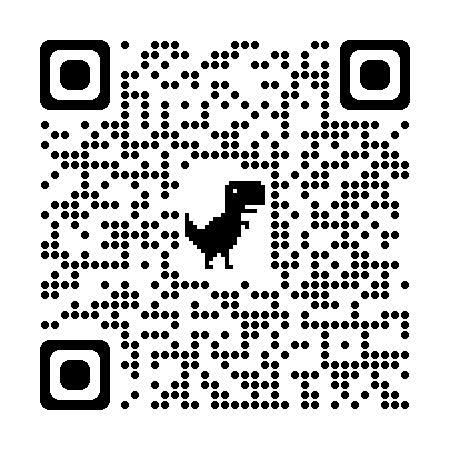
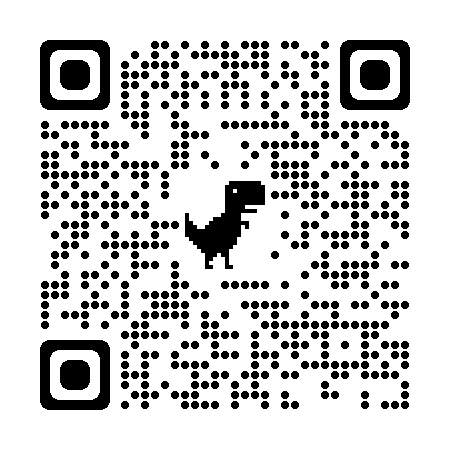
Description automatically generated

Advantages:

* No limit on the amount of letters the user can type
* Has a contact page to contact for any issues
* Login page to access previous creations
* 3D simulated image of the RNA strand created
* Instructions on how to input bases
* Able to save results by entering email and emailing results
* Able to save a text file for coordinates of each atom in the RNA molecule
* 2 modes for different uses
* Has a contact page to contact for any issues
* Has an instruction page on how it works (screenshots included here but not all instructions as it is long)
* Contains a link page connecting it to other related makers and databases

Disadvantages:

* Complex to use - as this is made by organisations aimed for researchers; this is not an appropriate teaching or revision tool
* Not user friendly - the UI is cluttered and the text is small even if zoomed in on
* Instructions on how to use is very long and not appropriate for students until they are Masters’ level
* Only loads on a certain browser - I had tried opening this website in Chrome but it did not work, so I tried Microsoft Edge and it worked
* No downloadable file to use locally incase the user cannot access the website anymore
* 3D simulation is slow to generate and spins to show all sides of the molecule but it spins faster at some angles and barely at all in others - inconsistent



#### *QR Codes for each website respectively: Nucleic Acid Converter, Protein Translator, and RNA Composer*

### Application Stack Research

Several analyses were done before finalising an application stack that would be best suited to build this application.

Some studies involved in analysis of using:

* Visualisation: Python, Javascript libraries like ViennaRNA (<https://github.com/ViennaRNA/forna>)
* Protein data banks (pdb) file analysis for RNA, Viztein (<https://github.com/thetechnocrat-dev/viztein>) that uses Javascript
* Protein data banks system stack analysis using Mol\* (<https://github.com/molstar/molstar>)
* Biopython python library, <https://biopython.org/DIST/docs/tutorial/Tutorial.html>
* RNA tools using rna-tools python library <https://pypi.org/project/rna-tools/>
* Research paper on build RNA structure, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7508704/>
* Understanding bioinformatics via python, <https://hplgit.github.io/bioinf-py/doc/pub/html/main_bioinf.html>
* Journal about using JNSViewer, <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0179040>
* Gene level exploratory analysis for RNA sequence that uses python and R, <https://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>
* Java libraries using VARNA for RNA secondary structure, <https://varna.lisn.upsaclay.fr/>

Based on the above research, it was evident that a collaborative approach was needed to view the information with uniformity. However, the basic building blocks relied on using Python to hold the bioinformatics-related libraries that possess almost all the features that would be required to be brought under a single application stack. Visualising the secondary structure from an application is also only partially available as a stand-alone that can be easily integrated.

### Solution and Why?

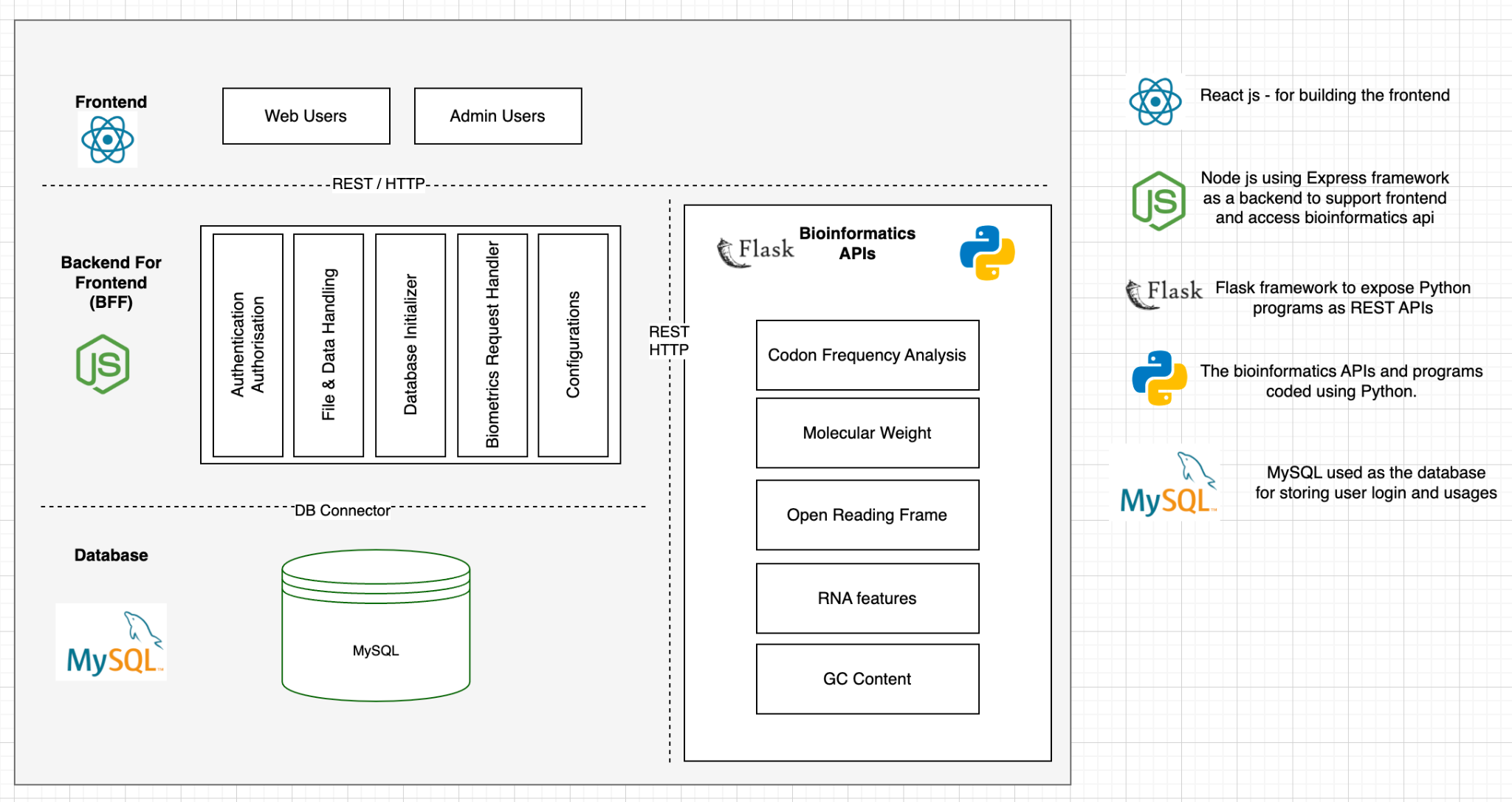
The complexity surrounding doing a bioinformatics analysis of an RNA sequence from the studies and research done, as mentioned under potential approaches, led to building a solution that:

* Would have a web **user interface** built using open-source libraries. This open-source should be lightweight and have proven communities and applications.
  + **React js** was chosen to develop the front-end application. It is javascript/typescript-based. Moreover, I have a very good understanding of the framework of React & Javascript full-stack application development.
* Have a decoupled **backend** for the front end that will hold all the business logic and be an abstract layer for the front end so that it focuses on building the user interface only.
  + **Node js** with Express framework was chosen to develop the backend. It is, again, lightweight and javascript-based. This can be run in a node js server. Moreover, I have a very good understanding of the javascript.
* Have a **freely available database that** can be installed, standalone or distributed.
  + **MySQL** was chosen in this case.
* Have a decoupled **REST API application** that serves RNA analysis in a single place.
  + **Flask and Python** were rightly chosen as they provide inbuilt and easy integrations with existing 3rd party Python libraries. Flask will be used to give the API application framework to expose the Python programs as REST APIs.

## 

# Application Stack

Below is a diagrammatic representation of the application and its components within each layer.



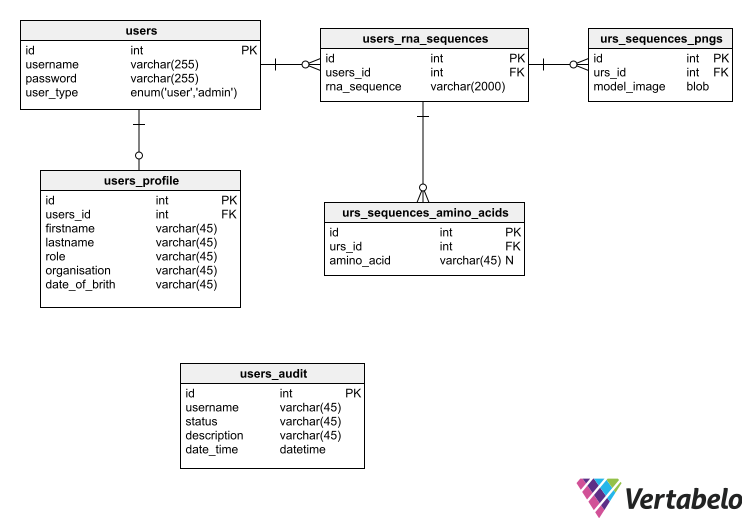
# Database Installation and Structure

The database that has been used in MySQL 8.0.

Installation of MySQL:

1. Download MySQL and follow the instructions at - <https://dev.mysql.com/doc/refman/8.0/en/installing.html>
2. Make sure you have provided a password for the ***root*** user.
3. Start the terminal and run the command, ***mysql*** ***-u <username> -p.*** This will ask the password that will connect to the MySQL server.
4. To list the databases, run command, ***SHOW DATABASES.*** This will list the installed databases.
5. The next step is to create the database for our application
   1. Creating the database: *CREATE DATABASE db\_rna;*
   2. Creating admin user: *CREATE USER 'db\_rna\_admin'@'localhost' IDENTIFIED BY 'password';*
   3. Grant all privileges to the admin user: *GRANT ALL PRIVILEGES ON db\_rna.\* TO 'db\_rna\_admin'@'localhost' WITH GRANT OPTION;*
   4. Create a db user that will be used to run default DML and DDL statements: *CREATE USER 'db\_rna\_user'@'localhost' IDENTIFIED BY '\*\*\*\*\*\*\*\*\*'*
   5. Grant DML and DDL privileges to the db user: *GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP ON db\_rna.\* TO 'db\_rna\_user'@'localhost';*

## Entity-Relationship Diagram (ERD)



## Data Definition Language (DDL)

CREATE TABLE `users` (

`id` int NOT NULL AUTO\_INCREMENT,

`username` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`user\_type` enum('user','admin') NOT NULL DEFAULT 'user',

PRIMARY KEY (`id`),

UNIQUE KEY `idx\_users\_id` (`id`),

UNIQUE KEY `idx\_users\_username` (`username`),

KEY `idx\_users\_id\_user\_type` (`id`,`user\_type`)

);

CREATE TABLE `users\_profile` (

`id` int NOT NULL AUTO\_INCREMENT,

`users\_id` int NOT NULL,

`firstname` varchar(45) NOT NULL,

`lastname` varchar(45) NOT NULL,

`role` varchar(45) NOT NULL,

`organisation` varchar(45) NOT NULL,

`date\_of\_brith` varchar(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

UNIQUE KEY `users\_id\_UNIQUE` (`users\_id`),

CONSTRAINT `fk\_users\_id` FOREIGN KEY (`users\_id`) REFERENCES `users` (`id`)

);

CREATE TABLE `users\_rna\_sequences` (

`id` int NOT NULL AUTO\_INCREMENT,

`users\_id` int NOT NULL,

`rna\_sequence` varchar(2000) NOT NULL,

PRIMARY KEY (`id`),

KEY `fk\_urs\_users\_id\_idx` (`users\_id`),

CONSTRAINT `fk\_urs\_users\_id` FOREIGN KEY (`users\_id`) REFERENCES `users` (`id`)

);

CREATE TABLE `users\_audit` (

`id` int NOT NULL AUTO\_INCREMENT,

`username` varchar(45) NOT NULL,

`status` varchar(45) NOT NULL,

`description` varchar(45) NOT NULL,

`date\_time` datetime NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `urs\_sequences\_pngs` (

`id` int NOT NULL,

`urs\_id` int NOT NULL,

`model\_image` blob NOT NULL,

PRIMARY KEY (`id`),

KEY `fk\_urs\_pngs\_id\_idx` (`urs\_id`),

CONSTRAINT `fk\_urs\_pngs\_id` FOREIGN KEY (`urs\_id`) REFERENCES `users\_rna\_sequences` (`id`)

);

CREATE TABLE `urs\_sequences\_amino\_acids` (

`id` int NOT NULL AUTO\_INCREMENT,

`urs\_id` int NOT NULL,

`amino\_acid` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `fk\_urs\_id\_idx` (`urs\_id`),

CONSTRAINT `fk\_urs\_id` FOREIGN KEY (`urs\_id`) REFERENCES `users\_rna\_sequences` (`id`)

);

To add:

* Human-computer/human-user interface - screenshots of UI
* System flow diagram
* Class diagram (inheritance)
* Data flow diagram
* Structure chart (what methods call other methods)/Hierarchy chart
* Algorithms in english or pseudocode - calculations such as protein prediction explained. Have already explained Zuker and Nussinov algorithms. **Database traversal has to be in pseudocode.**
  + Data structure
  + Data model
* **AGILE approach**
* **Prototyping - Hardware specification - minimum spec PC**
* Data dictionary
* File organisation
* Encryption algorithm description