Team name: dSign
Team members: Aurelia Leona, Zenia Valdiviezo, Laura Raiff, Sally Shin

Short Description: dSign is a web-based DNA signature encryption software. Users can create an account and input identifier information to create a signature in DNA. Then they can upload up to 5 DNA sequences to “sign”. There are two options for this: a reversible option (a highly randomized signature placement intended for computational work) and an irreversible option (intended to be sent straight to manufacturing where a type-II restriction enzyme is used to remove the signature during manufacturing). Finally, users can export and share their signed DNA sequences and the associated access key. The access key is sent to guest users to see information about the sequence and the designer.

Major Software Components:
User interface:

1. Inputted strings and pressed buttons are captured with HTML and saved with PHP. They are used to get the information needed to make the signature and signed sequence.
2. Components such as the Home and FAQ page make our software easy to use.
3. Javascript and CSS are used to make the appearance clean and appealing

Database:

1. PHP has session variables that enable the user’s data variables to be saved the entire time they are logged in and can use them across the pages. This functionality is essential to the existence of user accounts.
2. PHP sends SQL queries to a locally hosted database which stores each user's data in one row.
3. Detailed error checking interfaces with the database to prevent reducancy in data entry and to maintain a logical workflow of the user experience.

DNA and signature encryption:

1. PHP reads the signature and DNA sequence information from the database and writes them to JSON files. It also executes the following Python scripts:
    a. reads the JSON file with the signature information and translates the signature into nucleotides
    b. reads the newly created signature as a string and places the signature in the sequence based on the users specification (i.e reversible or irreversible)
    c. encrypts the sequence/message and the DNA signature and creates the public access key
2. The encrypted sequence and public access key are exported as shareable .key files

Guest access page:

1. PHP reads the names of user uploaded files from the HTML
2. PHP executes a Python script to decrypt the encrypted message DNA
3. PHP integrates with HTML so user can view the decrypted message on the web page

Encryption Explained:

Creating the signature:

1. PHP sends a JSON file to a python script with user inputted information.
2. Password characters are numbered 0-9, ordered based on ASCII values, and turned into an array of 10 columns.
3. The characters in user input information are appended as new rows to the password array in the order of last name, first name, institution type, institution code, and country.
    a. One row of information does not fit the full 10 columns, empty columns will be filled with 0s.
4. All characters are converted to the ASCII values and the sum of the columns is calculated.
5. Each sum is divided by the ASCII conversion of each nucleotide (A, C, G, T) such that there are four divisions.
6. The smallest remainder from four divisions will indicate which nucleotide will be designated for each of the 10 columns.
7. The signature is ordered by matching the order of the first row of the matrix with the original inputted password.

Placing the signature in the original sequence

1. Reversible: the translated signature's characters are placed separately in different randomized locations in the original sequence
2. Irreversible: the translated signature remains together and is placed in a user defined location flanked by a primer for the Type II restriction enzyme.

Encryption method for the keys:

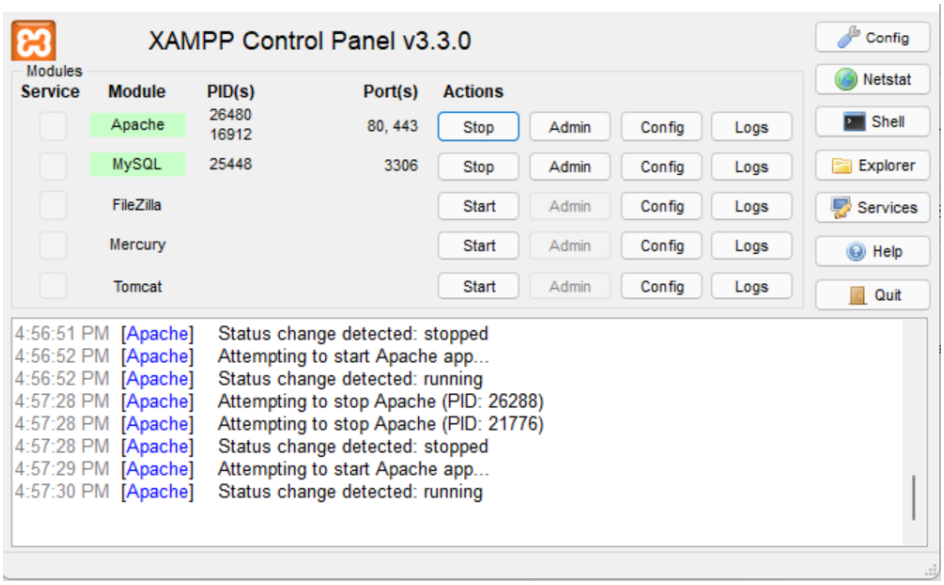
1. With the cryptography Python library, we used the Fernet type key, created with a hashing algorithm and a randomized salt
    a. This key is dependent on the inputted password to make the public access key that will be shared with another user
2. This code will encrypt a message that includes the name of the author of the sequence, their email address, the institution code and institution type, a description of the sequence, and the sequence with the signature embedded in it
3. This encryption code will generate a "public.key" (the public access key) and a "datacode.key" (the encrypted message) file
4. To decrypt this, there will be a need to input both files and the recipient of the files would be able to read the encrypted message

Special instructions when compiling the code (copy of the README):

1. Clone the repository on your machine: git clone https://github.com/lraiff/dSign.git
2. Install python (any version) and the following modules:

json	numpy	base64	cryptography
Fernet	hashes	PBKDF2HMAC	wrap

3. Install XAMPP and run the Apache and SQL servers at https://www.apachefriends.org/index.html. Find the xampp program file and the xampp-control application. The control panel should look like this:



4. Navigate to: **http://localhost/phpmyadmin/index.php** and create a database called "login" and a table called "users". In the table make the following columns:

users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	bigint(20)			No	*None*		AUTO_INCREMENT
2	**userID**	bigint(20)			No	*None*		
3	**user_name**	varchar(100)	utf8_general_ci		No	*None*		
4	**password**	varchar(10)	utf8_general_ci		No	*None*		
5	**date**	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()
6	**lastName**	varchar(20)	utf8_general_ci		No	*None*		
7	**firstName**	varchar(20)	utf8_general_ci		No	*None*		
8	**email**	varchar(50)	utf8_general_ci		No	*None*		
9	**institution**	varchar(5)	utf8_general_ci		No	*None*		
10	**instType**	varchar(10)	utf8_general_ci		No	*None*	type of instruction	

11	**encryptedSignature**	varchar(255)	utf8_general_ci		No	*None*		
12	**country**	varchar(30)	utf8_general_ci		No	*None*		
13	**Sequence1**	varchar(255)	utf8_general_ci		No	*None*		
14	**eSeq1**	varchar(255)	utf8_general_ci		No	*None*		
15	**designType1**	varchar(30)	utf8_general_ci		No	*None*		
16	**seqName1**	varchar(255)	utf8_general_ci		No	*None*		
17	**seqDesc1**	varchar(255)	utf8_general_ci		No	*None*		
18	**location1**	int(255)			No	*None*		
19	**Sequence2**	varchar(255)	utf8_general_ci		No	*None*		
20	**eSeq2**	varchar(255)	utf8_general_ci		No	*None*		
21	**designType2**	varchar(30)	utf8_general_ci		No	*None*		
22	**seqName2**	varchar(255)	utf8_general_ci		No	*None*		
23	**seqDesc2**	varchar(255)	utf8_general_ci		No	*None*		
24	**location2**	int(255)			No	*None*		
25	**Sequence3**	varchar(255)	utf8_general_ci		No	*None*		
26	**eSeq3**	varchar(255)	utf8_general_ci		No	*None*		
27	**designType3**	varchar(30)	utf8_general_ci		No	*None*		
28	**seqName3**	varchar(255)	utf8_general_ci		No	*None*		
29	**seqDesc3**	varchar(255)	utf8_general_ci		No	*None*		
30	**location3**	int(255)			No	*None*		
31	**Sequence4**	varchar(255)	utf8_general_ci		No	*None*		
32	**eSeq4**	varchar(255)	utf8_general_ci		No	*None*		
33	**designType4**	varchar(30)	utf8_general_ci		No	*None*		
34	**seqName4**	varchar(255)	utf8_general_ci		No	*None*		
35	**seqDesc4**	varchar(255)	utf8_general_ci		No	*None*		
36	**location4**	int(255)			No	*None*		
37	**Sequence5**	varchar(255)	utf8_general_ci		No	*None*		
38	**eSeq5**	varchar(255)	utf8_general_ci		No	*None*		
39	**designType5**	varchar(30)	utf8_general_ci		No	*None*		
40	**seqName5**	varchar(255)	utf8_general_ci		No	*None*		
41	**seqDesc5**	varchar(255)	utf8_general_ci		No	*None*		
42	**location5**	int(255)			No	*None*		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	5	A	No	
user_id	BTREE	No	No	userID	5	A	No	
user_name	BTREE	No	No	user_name	5	A	No	
date	BTREE	No	No	date	5	A	No	

5. Copy all the files in the cloned repo and place them a newly created folder with the path: ~/xampp/htdocs/dSign
6. Navigate to: http://localhost/dSign/