# CODE ANALYSER FOR BIOINFORMATICS

S.Aditya[1], N.Dinesh Reddy[2],Y.Srinath[3], Shamela Rizwana[4]

*[1] B.Tech Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India*
*[2] B.Tech Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India*
*[3] B.Tech Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India*
*[4]Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India*

## ABSTRACT

*"antiSMASH" is a great tool to predict secondary metabolite clusters. The software deals with large data coming in various forms such as GenBank, fasta etc. When dealing with such data it is important to write effective programs for faster processing.*

*"antiSMASH" uses Biopython's SeqRecord object to deal with the data. The current "antiSMASH" code has some drawbacks in the way it is implemented. This project proposal is to make an abstraction layer for "antiSMASH" to unify the access to different qualifiers/features entries. The existing code has some drawbacks by the way it is implemented. A lot of code duplication prevails in various parts of the code base. The new API layer will overcome the cons of the existing code and minimize the code duplication.*

**Keywords:** *-antiSMASH",Biopython, SeqRecord, API Layer, GenBank, Scemet*

## 1. INTRODUCTION

The **Biopython** Project is an open-source collection of non-commercial Python tools for computational biology and bioinformatics, created by an international association of developers. It contains classes to represent biological sequences and sequence annotations, and it is able to read and write to a variety of file formats. It also allows for a programmatic means of accessing online databases of biological information, such as those at NCBI. Separate modules extend Biopython's capabilities to sequence alignment, protein structure, population genetics, phylogenetics, sequence motifs, and machine learning. Biopython is one of a number of Bio* projects designed to reduce code duplication in computational biology.

This project proposal is to make an abstraction layer for "antiSMASH" to unify the access to different qualifiers/features entries. The existing code has some drawbacks by the way it is implemented. A lot of code duplication prevails in various parts of the code base. The new API layer will overcome the cons of the existing code and minimize the code duplication.

Significant part of the "antiSMASH" code base has to modified to adapt the new API layer .The entire input access is hidden behind this API layer. This not only improves the efficiency of software, but also provides easy access to various features and qualifier properties of the given input file.

At the end of the project, the standalone "antiSMASH" software will be restructured and rebuild upon this API layer minimizing the code duplication.

### 1.1 Existing and proposed systems

A lot of code duplication exists in various parts of the code base. The performance and efficiency can be drastically increased if we can reduce this code duplication. This can be achieved by introducing an API layer to unify the access to various features and qualifiers which can be obtained from the input files.

The project implementation happens in two phases:

1. Implementation of the abstraction layer.

2. Restructuring the current "antiSMASH" code to use the abstraction layer.

## 2.DOMAINS USED

### 2.1 "antiSMASH"

Bacterial and fungal secondary metabolism is a rich source of novel bioactive compounds with potential pharmaceutical applications as antibiotics, anti-tumor drugs or cholesterol-lowering drugs. To find new drug candidates, microbiologists are increasingly relying on sequencing genomes of a wide variety of microbes. However, rapidly and reliably pinpointing all the potential gene clusters for secondary metabolites in dozens of newly sequenced genomes has been extremely challenging, due to their biochemical heterogeneity, the presence of unknown enzymes and the dispersed nature of the necessary specialized bioinformatics tools and resources. Here, we present ""antiSMASH"" (antibiotics & Secondary Metabolite Analysis Shell), the first comprehensive pipeline capable of identifying biosynthetic loci covering the whole range of known secondary metabolite compound classes (polyketides, non- ribosomal peptides, terpenes, aminoglycosides, aminocoumarins, indolocarbazoles, lantibiotics, bacteriocins, nucleosides, beta-lactams, butyrolactones, siderophores, melanins and others). It aligns the identified regions at the gene cluster level to their nearest relatives from a database containing all other known gene.

### 2.2 SQLite

SQLite is an in-process library that implements a self-contained, server less, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite is a compact library. With all features enabled, the library size can be less than 500KiB.

### 2.3 RESTfull API

Representational state transfer (REST) or RESTful web services is a way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

An API for a website is code that allows two software programs to communicate with each another. A RESTful API explicitly takes advantage of HTTP methodologies. They use GET to retrieve a resource; PUT to change the state of or update a resource, which can be an object, file or block; POST to create that resource; and DELETE to remove it.

In computer programming, an **application programming interface** (**API**) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components.
A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, and computer hardware or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables or remote calls.

## 3. BLOCK DIAGRAM AND WORKING

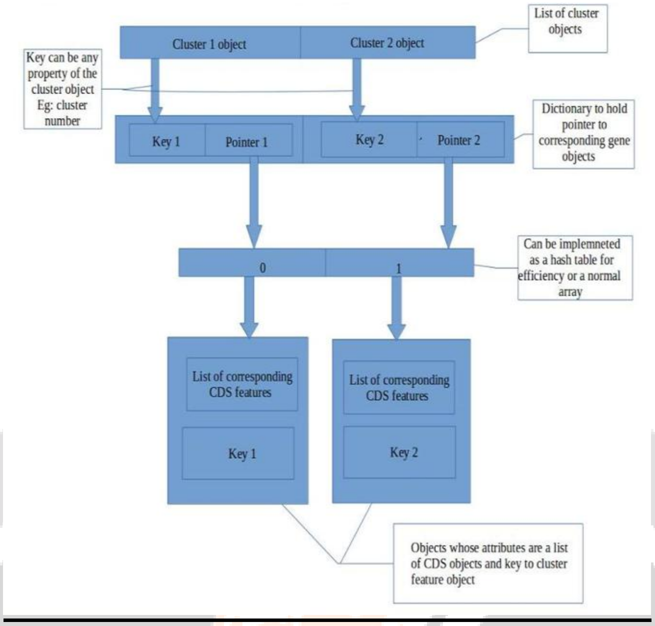

List of cluster objects
Cluster 1 object
Cluster 2 object
Key can be any property of the cluster object Eg: cluster number
Key 1
Pointer 1
Key 2
Pointer 2
Dictionary to hold pointer to corresponding gene objects
0
1
Can be implemneted as a hash table for efficiency or a normal array
List of corresponding CDS features
Key 1
List of corresponding CDS features
Key 2
Objects whose attributes are a list of CDS objects and key to cluster feature object

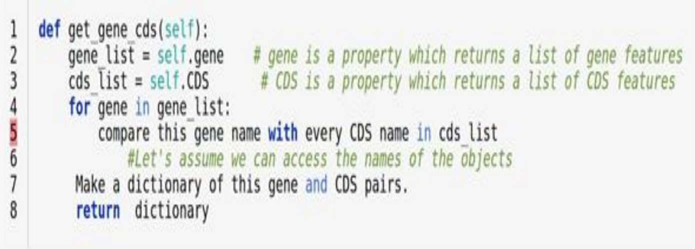
**Fig -1:**Block Diagram of antiSMASH layer

## 4.MODULES

- DESIGNING(Front End)- In this module we have done the coding basing on IEEE reference and through faculty support, we had uploaded our code in GitHub where every individual can take a look through our coding .the following is our website link:https://github.com/Manikumar1998
- BACK END- In back end procedure, we had downloaded many system requirements required by system to run the "antiSMASH" layer, and unfortunately this "antiSMASH" layer cannot be done in windows, so UNIX is preferable.

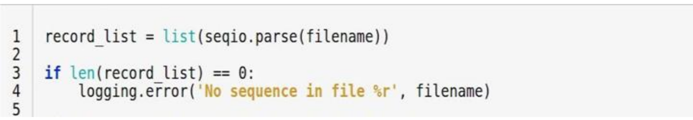
## 5. SAMPLE CODE

There are various ways of constructing or restructing the "antiSMASH" layer but it is to be coded in such a way that it should be restructed further , so that it can be remodeled in the future for any other use.



```
def get_gene_cds(self):
    gene_list = self.gene   # gene is a property which returns a list of gene features
    cds_list = self.CDS      # CDS is a property which returns a list of CDS features
    for gene in gene_list:
        compare this gene name with every CDS name in cds_list
            #Let's assume we can access the names of the objects
     Make a dictionary of this gene and CDS pairs.
     return  dictionary
```

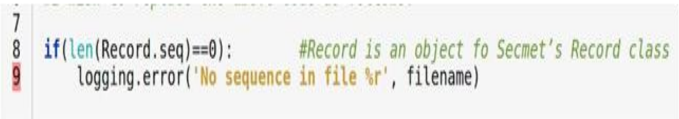
**Fig -2:P**rimary coding



```
record_list = list(seqio.parse(filename))

if len(record_list) == 0:
    logging.error('No sequence in file %r', filename)
```

**Fig -3:**Existing antiSMASH code



```

if(len(Record.seq)==0):       #Record is an object fo Secmet's Record class
    logging.error('No sequence in file %r', filename)
```

**Fig -4:**Proposed antiSMASH code

## 6. CONCLUSIONS ANDPERSPECTIVES

Thus basing on this project we strongly believe that we can reduce the code duplication using some coding based on Biopython and this would be helpful in many platforms like biotechnology and bioinformatics. “antiSMASH” tool plays a vital role in reducing the duplication of the code.

## 7. REFERENCES

1).Kai Blin, Marnix H. Medema, Daniyal Kazempour, Michael A. Fischbach, Rainer Breitling, Eriko Takano, & Tilmann Weber (2013): “antiSMASH” 2.0 — a versatile platform for genome mining of secondary metabolite producers. Nucleic Acids Research 41: W204-W212.

2)Marnix H. Medema, Kai Blin, Peter Cimermancic, Victor de Jager, Piotr Zakrzewski, Michael A. Fischbach, Tilmann Weber, Rainer Breitling & Eriko Takano (2011). “antiSMASH”: Rapid identification, annotation and analysis of secondary metabolite biosynthesis gene clusters. Nucleic Acids Research 39: W339-W346.

3) Department of Microbial Physiology and Groningen Bioinformatics Centre of the University of Groningen.

4)The Department of Microbiology of the University of Tubingen.

5)Department of Bioengineering and Therapeutic Sciences at the University of California, San Francisco