Computational and Structural Biotechnology Journal 27 (2025) 1001–1011

Contents lists available at ScienceDirect

Computational and Structural Biotechnology Journal

journal homepage: www.elsevier.com/locate/csbj

Software/web server Article

ProS^{2}Vi: A Python tool for visualizing proteins secondary structure

M. Luckman Qasim [a,b], Laleh Alisaraie [a,*]

[a] School of Pharmacy, Memorial University of Newfoundland, 300 Prince Philip Dr, A1B 3V6, St. John's, Canada
[b] Department of Computer Science, Memorial University of Newfoundland, A1C 5S7, St. John's, Canada

ARTICLE INFO

Keywords:
Proteins secondary structure
Python.
Folding
Local
Python

ABSTRACT

The **Pro**tein **S**econdary **S**tructure **V**isualizer (ProS^{2}Vi) is a novel Python-based visualization tool designed to enhance the analysis and information accessibility of protein secondary structures, calculated and identified using the Dictionary of Secondary Structure of Proteins (DSSP) algorithm. Leveraging robust Python libraries such as "Biopython" for data handling, "Flask" for Graphical User Interface (GUI), "Jinja2", and "wkhtmltopdf" for visualization, ProS^{2}Vi offers a modern and intuitive representation for visualization of the DSSP assigned secondary structures to each residue of any proteins' amino acid sequence. Significant features of ProS^{2}Vi include customizable icon colors, the number of residues per line, and the ability to export visualizations as scalable PDFs, enhancing both visual appeal and functional versatility through a user-friendly GUI. We have designed ProS^{2}Vi specifically for secure and local operation, which significantly increases security when working with novel protein data.

1. Introduction

Assigning secondary structures (SS) to proteins' primary structure (i.e., amino acid sequence) is a fundamental requirement for understanding their biological functions, stability, and intramolecular as well as intermolecular interactions. The secondary structures, comprising proteins' amino acids sequence, fold as α-helices, β-strands, β-turns, 3_{10} helices, etc., describe the secondary structure and, consequently, protein folding and functionality [1]. Visualization of the two-dimensional (2D) structures of amino acid sequences aids in the interpretation of complex molecular data and in explaining proteins' cellular, biophysical, biomechanical, and biochemical functions so that they can be interpretable to researchers and academic learners alike. Therefore, practical visualization tools are crucial for advancing our understanding of proteins' function as related to their physicochemical properties stemming from their small amino acid building blocks [2]. Visualization tools not only facilitate our understanding of protein structure but also enhance our ability to predict their interactions with other proteins, endo, and exogenous substrates, including therapeutical agents, which are pivotal, for instance, in drug discovery, medicine, and biotechnology [3].

Several algorithms have been developed to assist biochemists and biologists in studying proteins' secondary structures, each striving for greater accuracy — among the renowned algorithms such as STRIDE [4], KAKSI [5], and DSSP [6] — the latter often stands out as the foundation method. Kabsch and Sander introduced the Dictionary of Secondary Structure of Proteins (DSSP) in the early 1980s [6]. DSSP has remained a benchmark due to its rigorous methodology based on hydrogen bonding interaction patterns. DSSP classifies protein structures into eight types of secondary structures (e.g., helix, strand, H-bond turn, etc.), offering a detailed view that is critical for understanding protein folding and stability. Comparative studies have been conducted among DSSP and other algorithms such as STRIDE and KAKSI, especially concerning irregular peptides and complex structures — DSSP consistently shows robust performance in accurately assigning secondary structures [7]. However, the output from these algorithms is often textual and hard to interpret, which can be a significant barrier for those not specialized in bioinformatics. Visualization tools, therefore, play an important role in translating these textual outputs into graphical forms that are easier to understand and interpret.

Web-based visualization tools such as STRIDE web-server [8] and POLYVIEW-2D [9] are valuable for their accessibility and functionality. However, they face challenges that may render them less appealing in the contemporary context of protein research and cybersecurity. One of the main challenges is the security risks associated with these tools. A user is commonly required to upload a protein's structural information to an online web server to generate a 2D diagram to demonstrate the assigned secondary structure of a protein sequence or exhibit its folding in a linear format. STRIDE does offer a downloadable version; however,

* Corresponding author.
E-mail address: laleh.alisaraie@mun.ca (L. Alisaraie).

https://doi.org/10.1016/j.csbj.2025.02.038
Received 22 January 2025; Received in revised form 24 February 2025; Accepted 27 February 2025
Available online 28 February 2025

it only provides textual output, leading to the need to upload protein data to STRIDE's web-based version to produce visual output. Uploading unpublished data of a novel protein structure, a newly characterized structure either experimentally or computationally, to online web servers exposes sensitive data to potential cybersecurity threats, including unauthorized data access, data manipulation, and service disruptions caused by cyber-attacks such as Denial of Service (DoS) [10]. In general, these issues are compounded by the platforms' sporadic security updates, posing a significant risk when handling confidential or proprietary data [11]. Given these vulnerabilities, researchers must carefully evaluate the security policies and reputational trustworthiness of web-based tools. For those concerned with data security, alternatives such as local software solutions can provide more control over data security [12]. To address these concerns, we have developed a new local visualization tool using Python as a protein secondary structure visualizer (ProS^{2}Vi) [13]. The goal has been to address the security concerns associated with using web-based visualization tools for protein research and studying its secondary structure, folding, and detailed three-dimensional structure in a two-dimensional (2D) format.

ProS^{2}Vi [13] is a tool designed to operate entirely on the user's local machine. It significantly mitigates the risks of data breaches, data integrity attacks, and service disruptions commonly associated with online platforms. ProS^{2}Vi [13] not only enhances security but also offers a modern, user-friendly interface and detailed output that leverages the greatest in graphical technology, including modern-looking icons and high-quality output production for publications or presentations with accessible applications for non-technical biochemists, biologists, or for teaching protein science courses to students.

By running locally, ProS^{2}Vi ensures that all the protein data remains under the user's direct control without the need to transmit sensitive information over the internet. This setup is particularly beneficial for handling unpublished or proprietary research data concerning novel protein structures being under investigation *in vitro* using, for instance, protein crystallography and NMR techniques, *in silico* folded protein sequences, or novel bioengineered proteins, all of which require a high level of confidentiality prior to their publication. Furthermore, ProS^{2}Vi [13] is designed to be customizable, allowing researchers to adapt and expand its functionalities to meet specific needs, including the visual accessibility of 2D structure data. It supports various output file formats, such as Portable Network Graphics (PNG) [14], Joint Photographic Experts Group (JPEG) [15], and even Portable Document Format (PDF) [16].

A defining feature of ProS^{2}Vi is its ability to generate 2D visualizations of the secondary structure of a protein that also includes indices of each folded segment of the protein. (e.g., H1, for Helix 1, H2 for Helix2, …E1 for Strand 1, E2, Strand 2, etc.). Those are in addition to the protein's UniProt code [17] and the title of the paper publication associated with the protein structure on Protein Data Bank (e.g., RSCB PDB [18]), which are not shown in the output of other web-based tools, such as STRIDE [4] or POLYVIEW-2D [9] (Figure S1).

ProS^{2}Vi capabilities also set it apart from widely used tools such as PyMOL [19], Jmol [20], or Chimera [21]. These tools are excellent at rendering 3D molecular structures; however, unlike ProS^{2}Vi, they do not focus on 2D representations of protein secondary structure and folding. ProS^{2}Vi offers a simplified, highly informative view of protein secondary structure that includes annotations of α-helices, β-strands, turns, bends, 3–10 helices, etc., to enable researchers to quickly view the entire protein pattern without the complexity of 3D models.

2. Tool description

ProS^{2}Vi [13] is a 2D secondary assignment depicter tool developed using Python and works using the DSSP algorithm [6] output. It depicts the secondary structure of a protein based on the standard protein folding representations, including α-helices, β-strands, β-bridges, 3_{10}-helices, H-bonded turns, etc., according to the pertinent descriptions of the DSSP algorithm [6].

2.1. Implementation

ProS^{2}Vi [13] accepts protein structure with a Protein Data Bank (PDB) [18] "pdb" file format or a macromolecular Crystallographic Information Framework (mmCIF) [22] file. It uses Biopython [23] and the DSSP executable file to extract the DSSP output pertaining to the structure of the protein of interest. The DSSP executable file can easily be downloaded on Linux or Windows Subsystem for Linux (WSL) using the standard Advance Packaging Tool (APT) [24], which is the default user interface working with core libraries to handle the installation and removal of software on Linux distributions. Next, the DSSP output is formatted in the form of a Python dictionary, where each element represents a separate protein chain — within each protein chain element is a Python list containing amino acid residues and their relevant secondary structure predictions, according to the DSSP output. Once this Python list is ready and includes all the required information about the protein chain identification, amino acid residues, and their relevant secondary structures, it is used to populate a HyperText Markup Language (HTML) table using "Jinja2" [25]. "Jinja2" is one of the fastest templating engines, which can be used to create HTML documents using Python. Since protein structures can often be quite large, a fast-templating engine is necessary. The result is an HTML table, divided into separate parts by the chains of a multi-subunit protein structure (Fig. 1).

The HTML table contains the protein structure file name or its "pdb" ID from Protein Data Bank, a short description of the protein type, and the organism. The information in the table is followed by the protein's chain ID and its sequence ID from UniProt (i.e., the resource for the protein sequence) [17], as well as other additional protein functional information. Each part of the table consists of three alternating rows: *i.*, the first row contains secondary structure indices counting the occurrences of α-helices, β-strands, β-bridges, 3_{10}-helices, π-helices, H-bonded turns, and simple bends, such as "H1" for the first occurrence of α-helix, "H2" for the second occurrence of α-helix, etc. *ii.* The second row contains the secondary structural icons for residues, exhibiting a coil for α-helices, 3_{10}-helices, and π-helices with different color representations, an arrow for β-strands and β-bridges, and cylinders of varying thickness for H-bonded turns, simple bends, and unsolved structures and *iii.* The third row consists of single-letter coded amino acids of the protein sequence (Fig. 2A).

Once the HTML table is created, it is converted into a PDF or rendered as an image. This is carried out via an open-source tool, "wkhtmltopdf" [26], to render HTML documents to PDF and various other image formats. "Wkhtmltopdf" is an open-source command-line tool written in C++ language. Python libraries (e.g., "PDFKit" [27] and "IMGKit" [28] can be utilized to function as wrappers, allowing the use of such tools (Fig. 1).

As an example, ProS^{2}Vi can be utilized to study changes in protein folding/unfolding in different states of proteins' conformations in the RRM Domain of ETR-3 in 2MY7[30]. ETR-3 is an RNA-binding protein acting as an alternative splicing factor that is involved in the stability and regulation mechanism of certain RNA messengers [30]. When ETR-3 is abnormally expressed, it can result in health complications such as progressive muscle wasting process in myotonic dystrophy. Thus, a detailed evaluation of the protein's stability, as relates to its folding/unfolding mechanism, is vital to explain the protein's mechanism of action and explore adequate biomedical or pharmaceutical approaches to regulate its expression and function (Fig. 2B).

ProS^{2}Vi can also be used to study the folding/unfolding of protein conformations generated using computational calculation methods such as molecular dynamics (MD) simulations. For instance, an MD simulation-resulting conformation of a main building block of Microtubule, an essential cytoskeletal protein (5JCO[31]), was analyzed using ProS^{2}Vi. Dynamic conformational changes of α, β-tubulin heterodimer

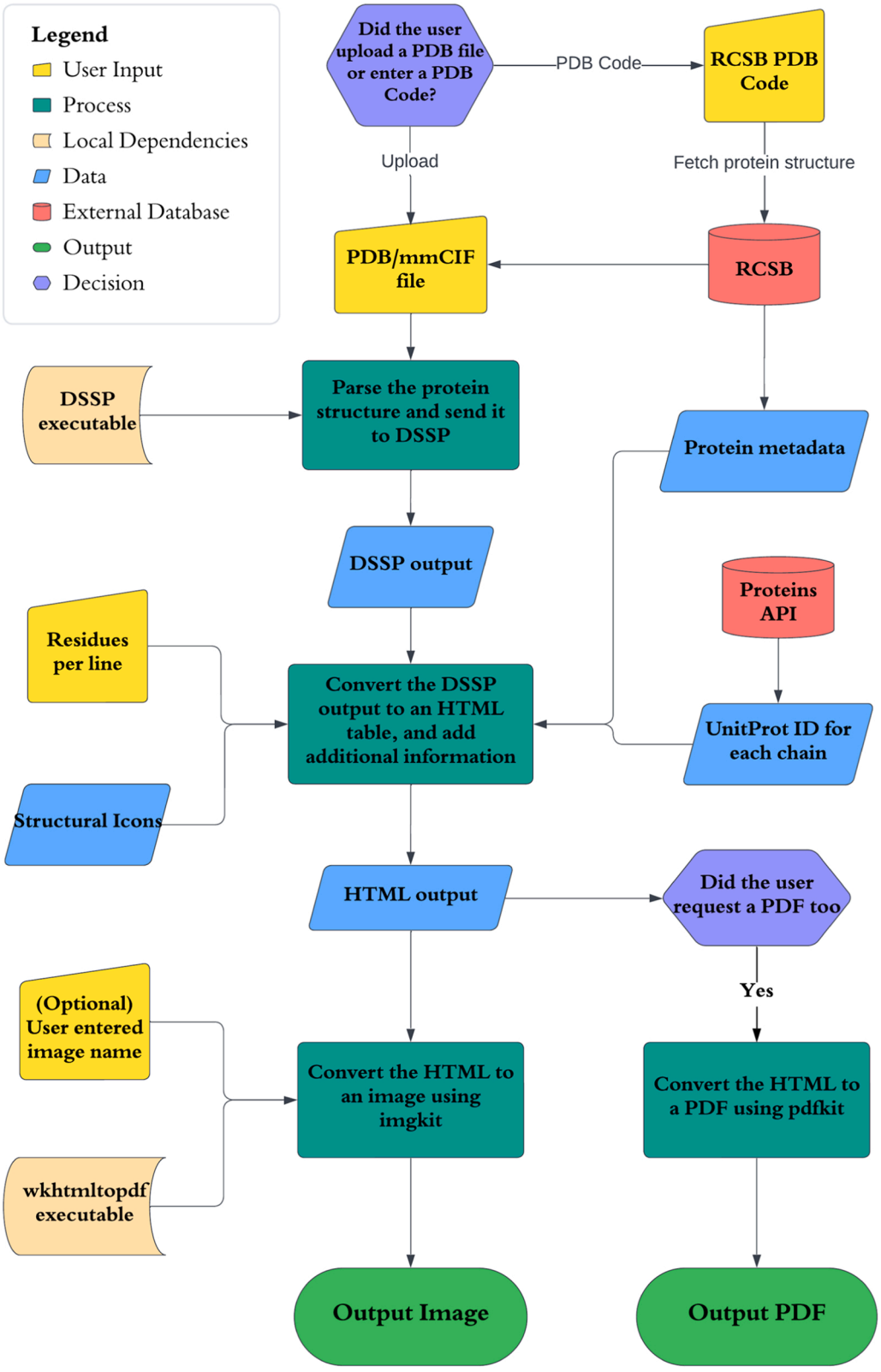

Fig. 1. The flowchart summarizes ProS^{2}Vi's [13] roadmap from the input structure file entry to the generation of the output image.

were studied using GROMACS [32], a well-known and widely used MD simulation package — this indicates the applicability of ProS^{2}Vi to analyze the results of *in silico* calculated or simulated protein atomistic coordination data (Fig. 2C).

2.2. Requirements

A Linux machine is preferred to use ProS^{2}Vi [13] since its dependencies, including DSSP and "wkhtmltopdf," are much more straightforward to install on Linux than on Windows. To run this on a

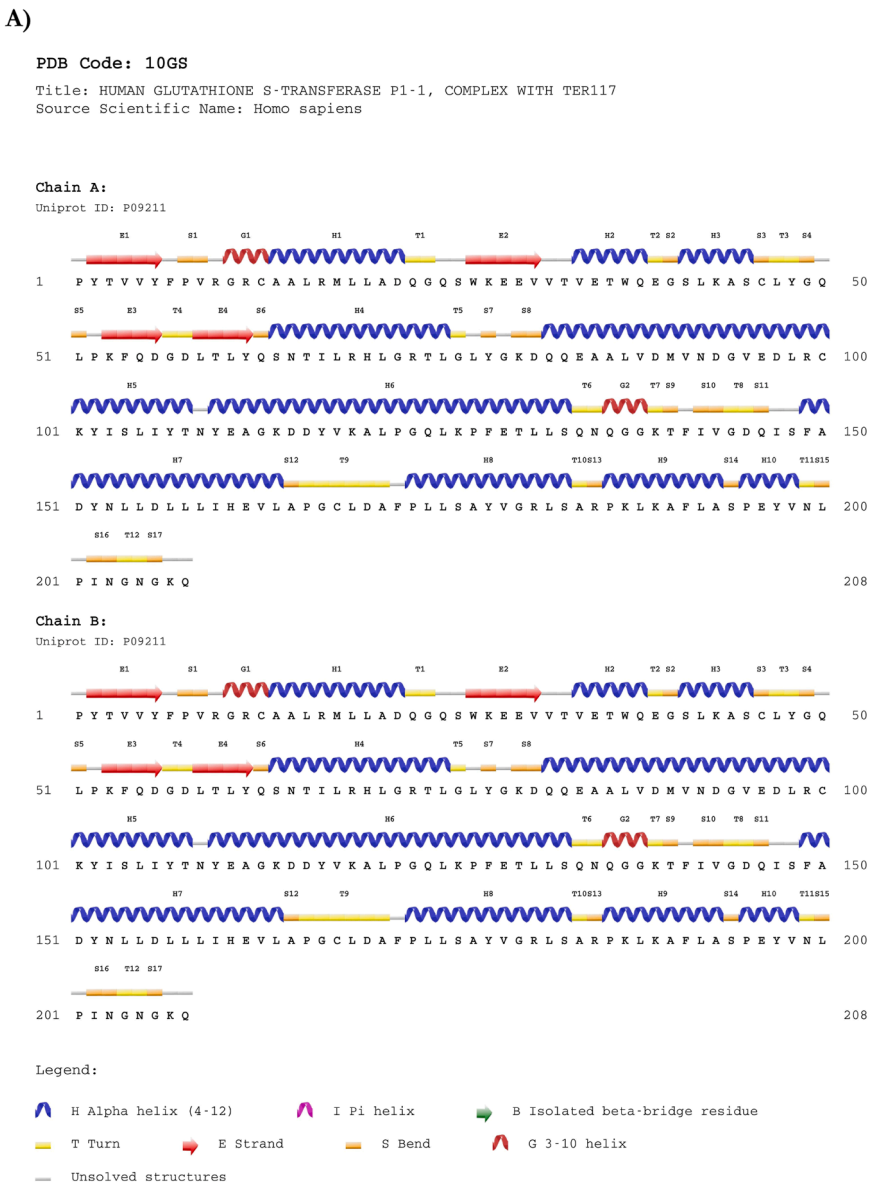

Fig. 2. (A) An example presenting the visualization for protein structure 10GS [29], produced by ProS^{2}Vi [13] using default settings. The output visualization clearly shows the file name or protein's PDB code, along with a short informative description of the protein and its sourced organism. All the chains in the multi-subunit protein structure appear along with their amino acid sequences pertaining to the protein identification code (ID) in the database of protein sequences (i.e., UniProt), and relevant secondary structure icons on the top row, as predicted by the DSSP algorithm. This example was generated with default settings and icon colors.* (B) Three conformations of the ETR-3 C-terminal RRM domain were extracted from the NMR structure with a total of 20 different conformations in 2MY7 [30]. ProS^{2}Vi result is shown next to the pertinent conformation of the tertiary structure, characterized using the NMR technique, as it can be seen using the 3D visualization software PyMOL [19]. The presented conformations are from frame 1 (green), frame 11 (cyan), and frame 15 (magenta) of the NMR data in 2MY7[30]. ProS^{2}Vi clearly and in detail presents the changes in the folding of the protein in each ETR-3's RRM conformation. (C) A conformation extracted from a Molecular Dynamics Simulation trajectory of α, β-tubulin heterodimer (5JCO.pdb [31]), obtained using the GROMACS package [32]. *: *Additional options are readily provided to change the number of residues in each row or the color of the icons using the color palette in GUI (See Fig. 4).*

Windows machine, a Windows Subsystem for Linux (WSL) [33] is preferred. WSL allows the users to run a Linux environment on Windows without the need for a separate virtual machine or dual booting. WSL can be downloaded free of charge from Microsoft's official website and is easy to install. Once the environment is set up, the repository containing the code needs to be cloned, and the dependencies need to be installed. The required dependencies to run ProS^{2}Vi [13] include Python, DSSP, "wkhtmltopdf," and all other required Python libraries, including "Biopython," "Jinja2", "IMGKit," "PDFKit," "Flask" [34], "pdf2image" [35], "CairoSVG" [36], "Pillow" [37], and "Requests" [38]. All the Python dependencies are listed in the "requirements.txt" file in the git repository and can be easily installed using the Python package installer (pip). While these dependencies, along with non-Python dependencies like DSSP and "wkhtmltopdf" can be installed manually, we have included a shell script in the package to simplify the installation process for the average user. This shell script examines whether all the required dependencies exist in the system; otherwise, it installs them, making the entire installation process automatic and much more straightforward than manually installing all dependencies individually. Once all the requirements are installed, ProS^{2}Vi [13] can be run using either the basic command-line interpreter or the more user-friendly Graphical User Interface (GUI) designed for this SS depicter (Fig. 3).

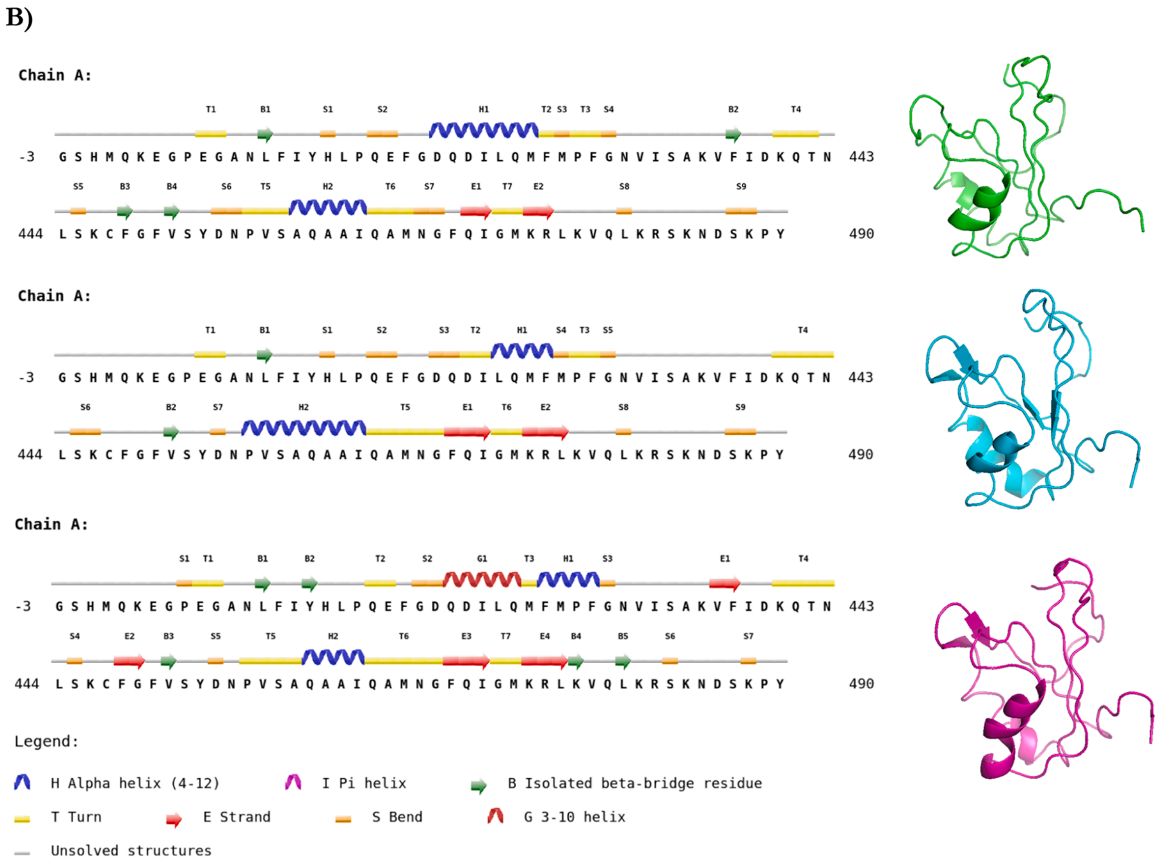

Fig. 2. (*continued*).

2.2.1. Difference between Linux and Windows installation

The installation of ProS^{2}Vi [13] on Linux versus Windows is primarily identical, except for the fact that for Windows, the users first need to download and install WSL, following the steps on Microsoft's official website [33]. Once that is installed, the user can open the WSL terminal, and the next steps are the same as those for Linux. The next step is cloning the git repository containing the software and installing all the required dependencies, either manually or automatically, by running the commands "chmod +x install_dependencies.sh" and "./install_dependencies.sh". Next, the user can use ProS^{2}Vi's [13] GUI by running the command "python3 pros2vi_gui.py" on the terminal, which should open a web browser in Linux. In WSL, users might need to manually open their web browser and enter "http://127.0.0.1:3000" to access the GUI.

2.3. Graphical user interface and customizability

To use the GUI, a Python file named "pros2vi_gui.py" should be run using the terminal, which then launches an instance of a web browser showing the interface. If the script runs on WSL, the browser might not launch automatically. Instead, it needs to be manually opened by opening a browser and entering the address shown in the terminal subsequent to running the script. It is noteworthy that despite the GUI running on a web browser, it is by no means connected to the internet (Fig. 4).

ProS^{2}Vi [13] has two input options: "Look-up PDB File" and "Upload PDB File." The first option accepts the protein structure code from the protein data bank (PDB) and downloads the file from the database of the "Research Collaboratory for Structural Bioinformatics" (RCSB) [39]. The second option requires the user to upload a file in either "pdb" or the "mmCIF" file format containing the atomic coordination data of any protein's structure. This option is helpful for generating visualization for novel protein structures that are not yet available on RCSB. With this option, the users have to manually enter the "Title/PDB Code," "Subtitle," and "Source Scientific Name" of the protein structure. If any of these input boxes are left empty, then the related heading is omitted from the output, providing users with the ability to generate visualizations with no headings.

Once the required structural data inputs are selected, the user can customize the ProS^{2}Vi [13] output by entering information in the optional fields. The customizations include the ability to change the number of amino acid residues that appear per line, which defaults to 50. The user can also change the file name of the output image and its format, where the supported formats include JPEG and PNG. If no input is provided in this field, the image name defaults to the protein structure code in a PNG output file format. Furthermore, the ability to easily change the color of structural icons provides flexibility and customization to suit individual needs and preferences. These colors can easily be changed for individual icons by pressing the relevant icon button in the GUI, which then opens a color palette, providing the ability to choose any of the available colors (Fig. 4).

The command line interface (CLI) version of this script is less flexible since it does not support the ability to change the colors of the structural icons or access the protein structure files from RCSB; however, it can be helpful for technical users because it can be coupled with a shell script to generate batch visualizations. For the CLI, the default settings can easily be modified by providing optional arguments when running the script. More information about the CLI positional arguments can be seen by running the CLI with a "–h" parameter, which represents "–help" and lists all available commands.

The default settings currently produce a PNG image with a 100 dots per inch (DPI) resolution. However, the resolution (DPI) can easily be changed in both GUI and CLI versions. It is noteworthy that increasing the DPI might significantly increase the processing time for generating

C)

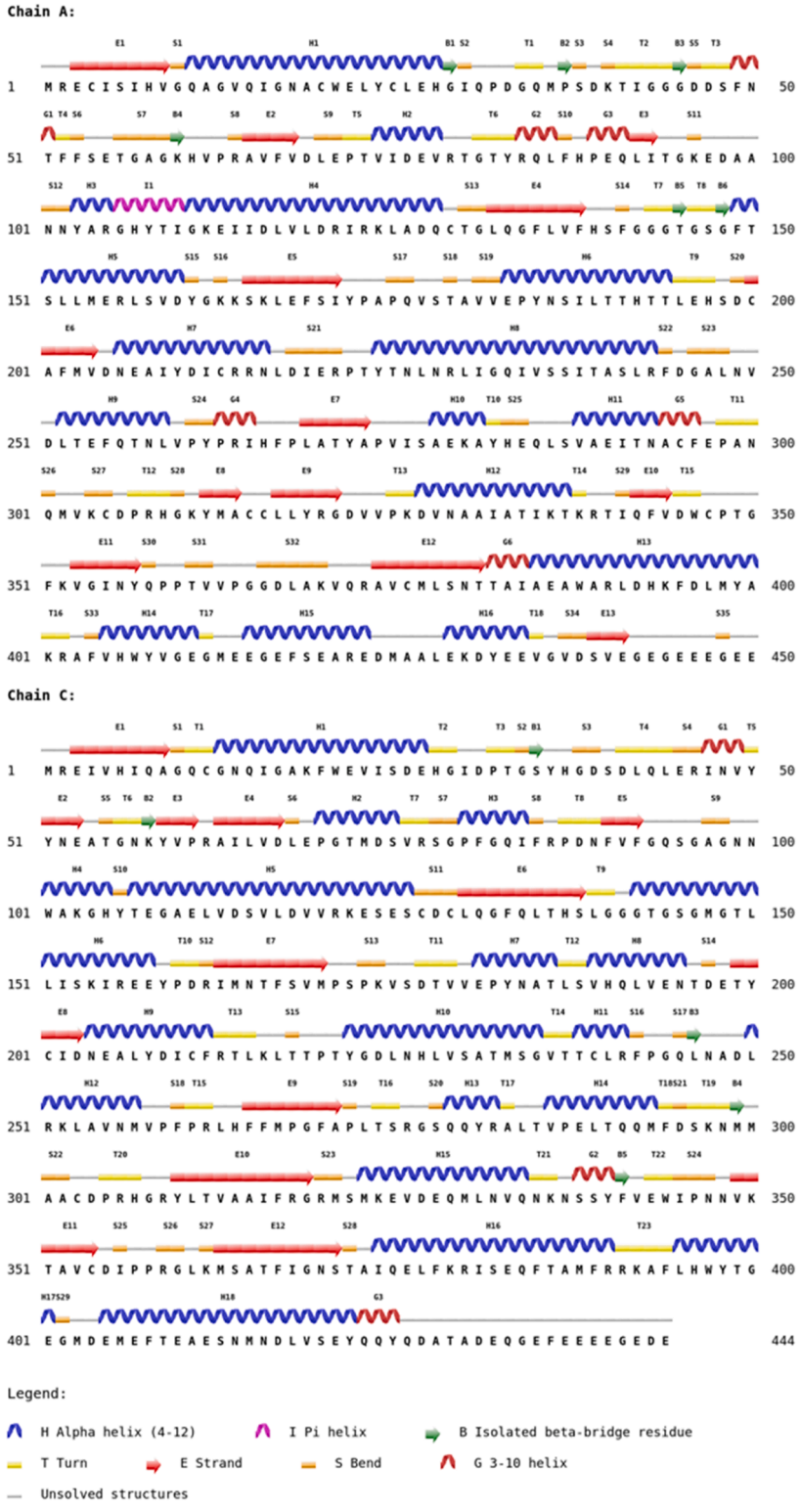

Fig. 2. (*continued*).

the visualization or might not work for significantly larger protein structures containing several thousand modeled residues. However, to create a higher resolution for larger structures, a PDF version of the visualization can be produced.

3. Assessment

The development of a Python-based visualization tool for protein secondary structures yields several vital outcomes that significantly enhance the analyses and presentation of protein data. We evaluated the performance of ProS^{2}Vi's [13] on different protein structures with varying residue counts and chain numbers using "pdb" and "mmCIF" file

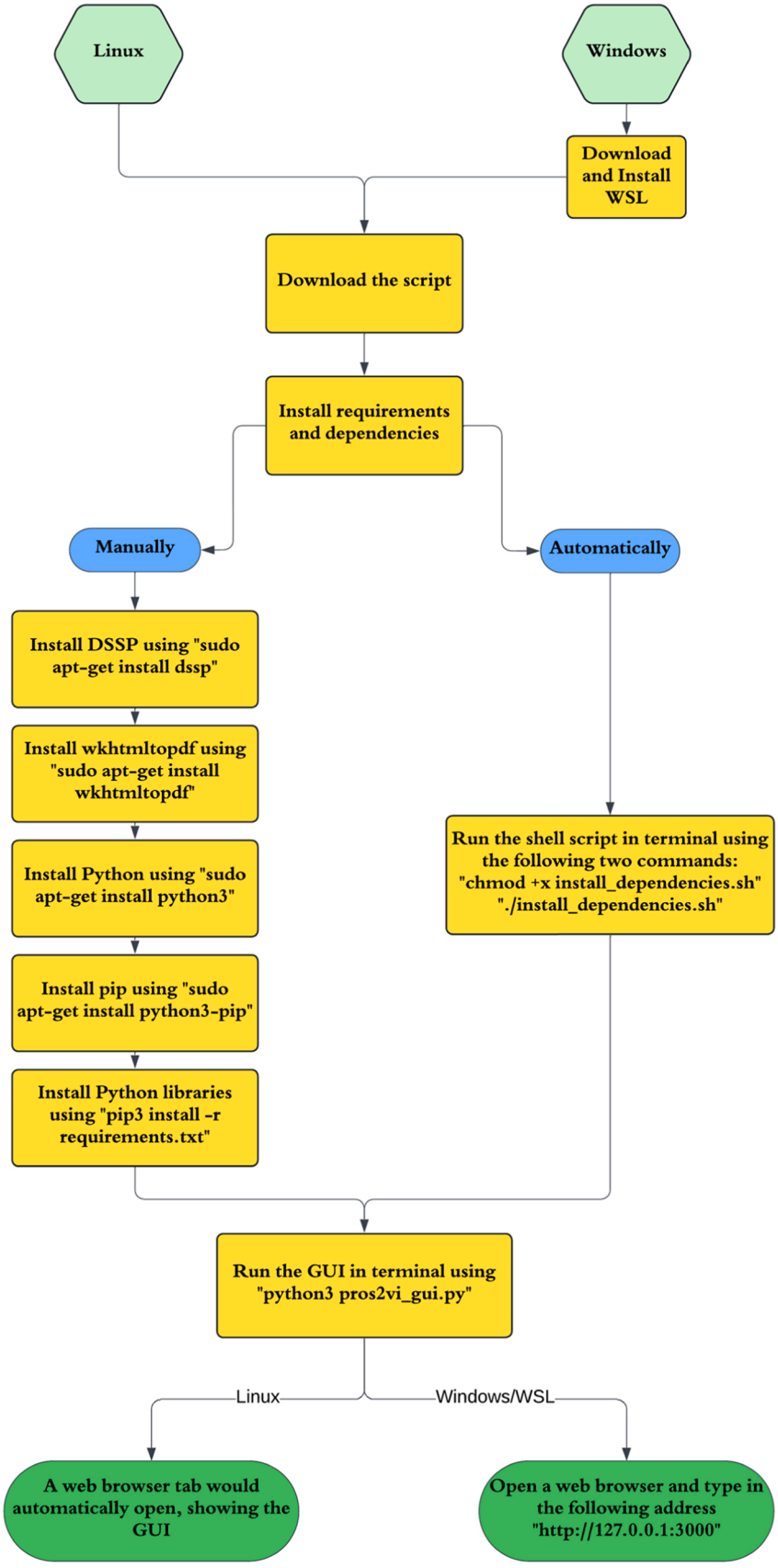

Fig. 3. The flowchart summarizes the entire installation process for ProS^{2}Vi [13] on both Linux and Windows machines. *The use of shell script makes the entire installation process much easier compared to manually installing dependencies.*

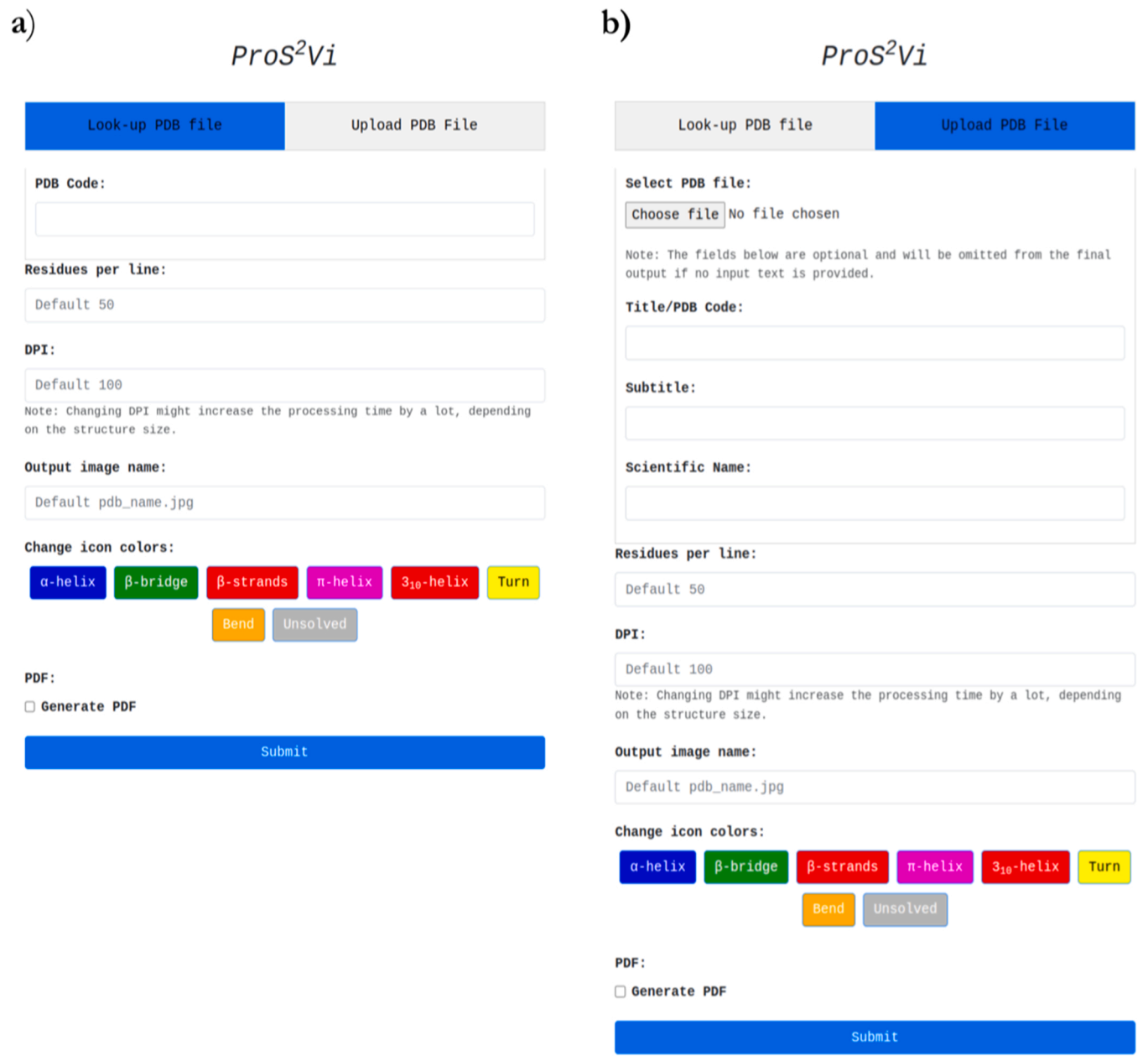

Fig. 4. The Graphical User Interface (GUI) of ProS^{2}Vi [13], demonstrating its layout and the available options. The blue tab at the top shows the current input selection. A) the "Look-up PDB file" option can be seen, which only requires an RCSB PDB code to generate visualization. B) the "Upload PDB File" option, which allows the user to upload their own PDB file and add their own "Title," "Subtitle," and "Source Scientific Name." *If either of the "Title," "Subtitle," or "Source Scientific Name" is left empty, then that heading is omitted from the visualization.*

formats. The following sections provide a detailed analysis of the ProS^{2}Vi [13] performance metrics and visualization capabilities.

3.1. Extraction and visualization accuracy

Assessing ProS^{2}Vi [13] showed that it effectively extracted the required information from the input structural files, such as amino acid sequences and their predicted DSSP-based secondary structure. It demonstrated robust performance across various protein structures, accurately presenting secondary structures in a 2D format (e.g., α-helices, β-strands, and other motifs) using distinct icons and colors. This ability to differentiate between various secondary structures is crucial for understanding protein folding and function. The results showcase the capability of ProS^{2}Vi [13] to generate precise, high-quality, and detailed visualizations that are easy to interpret. For instance, it allows users to quickly identify structural elements and their spatial relationships within the protein using different colors and icons for various secondary structures. This feature is particularly beneficial for educational purposes and presentations, where self-explanatory, simple, and intuitive visualizations are essential.

3.2. Performance metrics

The efficiency of ProS^{2}Vi [13] was evaluated using various protein structures with varying sizes and complexities (e.g., multi-subunit, multi-chain). The performance metrics were recorded for both "pdb" and "mmCIF" file formats and included parsing and extracting DSSP output, creating HTML tables, and converting HTML to images (Table 1).

The time required to parse and extract DSSP output varied

Table 1
Average run time (in seconds) for various protein structures, using both "pdb" and "mmCIF" protein structure file formats. *These tests were performed on a Linux machine with an Intel Core i5-4570 CPU and 16.0 gigabytes of memory.*

PDB Code& sequence length (AA)	Parsing and extracting DSSP output (s)		Creating HTML table (s)	Converting HTML to image (s)	Total time (s)
1Q7O					
3	pdb	0.441	0.035	0.328	0.805
	mmCIF	0.241	0.034	0.323	0.597
4DX9					
3767	pdb	6.437	4.283	0.545	11.264
	mmCIF	4.310	4.223	0.535	9.068
3KGV					
4064	pdb	10.601	11.098	0.790	22.489
	mmCIF	9.447	10.608	0.748	20.804
2VDC					
11,568	pdb	30.481	32.943	2.295	65.721
	mmCIF	23.894	31.749	1.679	57.323
8RJK					
19,548	pdb	58.619	28.208	1.510	88.346
	mmCIF	53.970	27.914	1.487	83.371
8VRJ					
24,650	pdb	-	-	-	-
	mmCIF	91.443	29.923	1.658	123.025

significantly between the file formats. For instance, the DSSP parsing time for the "pdb" file of a protein with 8RJK retrieving code [40] was ~58.619 sec, whereas its "mmCIF" file format took ~53.970 sec — it indicates that "mmCIF" files generally allow faster parsing. The time taken to create HTML tables and convert them to images also varied; however, to a lesser extent. For example, creating an HTML table of a protein with the PDB 2VDC retrieving code [41] took approximately 32.943 sec for the "pdb" file format and 31.749 sec for its "mmCIF."

The conversion of HTML to images was relatively rapid, taking nearly 1.679–2.295 sec across different files. The total time to process and generate visualizations ranged from a few seconds for a short peptide such as 1Q7O [42] to over a minute for a multi-chained system such as 8RJK [40]. The faster processing time for "mmCIF" file format highlights the advantage of using this format for large datasets (Table 1).

ProS^{2}Vi also demonstrates efficient memory management across various stages of processing. For instance, with 10GS [29], with 416 amino acids count, the total memory increase is only 9.43 MB, of which 4.9 MB is consumed for parsing the structure and producing DSSP output, indicating the tool's low overhead for less complex structures. Similarly, for 3KGV [43] with a 4064 residue count and 2VDC [41] with an 11,568 residue count, the total increases are 53.40 MB and 56.46 MB, respectively. The most significant memory increase is observed with 8VRJ, with a rise of 261.66 MB. This memory consumption is justified due to the 24,650 residue count of 8VRJ.

3.3. Case study examples

As mentioned, the visualization for 8RJK, a large protein with 19,548 modeled residues, demonstrated ProS^{2}Vi's [13] robustness in processing complex datasets. The total processing time for the PDB file was ~88.346 seconds, with most of the time (~58.619 seconds) spent on DSSP parsing. The "mmCIF" file processed faster at ~83.371 seconds, showcasing the efficiency gains with this format. Furthermore, the visualization for 8VRJ [44], the largest size protein structure, in terms of its number of amino acids or atoms available to test, with 24,650 modeled residues, required ~123.025 seconds to process, with most of the time (~91.443 seconds) taken by DSSP calculation. This test further proves the efficiency of our code, considering the sheer size of the protein structure. For a short peptide such as 1Q7O, with only a few characterized residues, the tool processed the PDB file in ~0.805 seconds and the mmCIF file in ~0.597 seconds. This case highlights the ability of ProS^{2}Vi [13] to treat minor-size datasets quickly; however, it is also well-functional for a wide range of protein sizes. The results demonstrate that the Python-based visualization tool is highly effective in extracting, processing, and visualizing protein secondary structures. Its performance across different protein sizes and formats showcases its versatility and robustness.

3.4. Security and accessibility

ProS^{2}Vi [13], operating locally, eliminates the risks associated with uploading sensitive data to online servers, ensuring the confidentiality and integrity of proprietary research data. That is essential for handling unpublished novel protein structures. Local operation means that sensitive data remains on the user's machine, reducing the risk of exposure to cyber threats that are prevalent on web-based platforms. Moreover, ProS^{2}Vi runs entirely on local hardware; thus, there is no interruption due to server downtime or online service interruptions, which is different from web-based tools. That ensures consistent availability and reliability of ProS^{2}Vi for users.

3.5. Performance and file compatibility

The tool supports multiple file formats, including "pdb" and "mmCIF" ensuring broad compatibility with existing bioinformatics workflows and databases, whereas STRIDE and POLYVIEW-2D only process files in "pdb" format. This flexibility allows users to work with various data sources without the need for extensive data conversion. The ability to process both file formats is crucial because it will enable researchers to choose the format that best suits their needs, considering that "mmCIF" files often provide slightly faster parsing times than their equivalent "pdb" counterparts (Table 1).

The length of a protein's amino acid sequence can significantly impact the software's performance time, potentially increasing processing times for extensive datasets. That is an essential consideration for researchers working with massive structural data that can affect the tool's efficiency and the ability to deliver fast output. ProS^{2}Vi [13] was initially created using "WeasyPrint" [45] and "pdf2image" instead of "wkhtmltopdf," "PDFKit" and "IMGKit" to eliminate the dependency on "wkhtmltopdf", and to allow it to generate images with higher DPI. This implementation was later changed to use "wkhtmltopdf" instead because "WeasyPrint" and "pdf2image" took a considerable amount of time to convert the HTML table to a PDF or image, which increased significantly with the proteins with a greater number of amino acids or more significant structure size. That was tested by comparing the two implementations and the time required to visualize specific protein structures using the "mmCIF" file format. The time necessary for "WeasyPrint" is significantly higher than that for "wkhtmltopdf." For 3KGV [43], which contains 4064 modeled residues, "WeasyPrint" takes around 245.437 seconds to convert the HTML to an image, whereas "wkhtmltopdf" takes only ~0.748 seconds, which shows a significant improvement (Table 2).

3.6. Limitations

While ProS^{2}Vi [13] offers significant advantages, there are limitations to consider. Currently, the tool can effectively process protein

Table 2
The comparison between the necessary time for "WeasyPrint" and "wkhtmltopdf" (in seconds) to convert an HTML table to an image.

PDB Code	Time to convert HTML to Image (s)	
	Using "WeasyPrint" and "pdf2iamge"	Using "wkhtmltopdf" and "imgkit"
1Q7O	0.458	0.323
4DX9	70.892	0.535
3KGV	245.437	0.748

structures with up to ~25,000 modeled residues due to the restrictions imposed by the ability of the DSSP algorithm and Biopython. This constraint means that extremely large-sized proteins may require alternative methods for secondary structural assignments. Additionally, the faster processing times for ProS^{2}Vi [13] currently only work when producing images with a 100 DPI resolution due to the limitations of "wkhtmltopdf". When creating an image with a different DPI, the algorithm first produces a PDF and then converts it into an image using "pdf2image" instead of "IMGKit," which slightly increases the processing times for smaller protein structures. However, for much larger protein structures, the time increases significantly, and some significantly large structures (>~6000 modeled residues) might not produce any output at all, depending on the machine specifications. Addressing these limitations in future versions of the tool will be a priority to enhance its applicability and performance further.

The processing time of DSSP and its memory consumption directly affect the overall time efficiency of ProS^{2}Vi, particularly for large and complex multi-domain protein structures. Future development efforts will be focused on optimizing internal processes and improving efficiency. However, the inherent performance of DSSP will continue to be an essential factor in overall speed and memory usage for ProS^{2}Vi.

3.7. Comparison with STRIDE and POLYVIEW-2D

While some existing online tools, such as STRIDE and POLYVIEW-2D could perform faster for specific large-size proteins, ProS^{2}Vi demonstrates competitive performance, especially when considering the test environment. The tests were conducted on a Linux machine with an Intel Core i5–4570 CPU and 16 GB of RAM — for example, the processing time for a protein from the RCSB Protein Data Bank, 8RJK consisting of 96,701 atoms, required ~43 sec time using STRIDE, ~32 sec with POLYVIEW-2D, and ~88 sec using ProS^{2}Vi, which included ~58 sec for the DSSP algorithm to run. However, for another protein from the RSCB data bank, 3KGV with 20,333 atoms, STRIDE required ~45 sec, POLYVIEW-2D ~43 sec, and ProS^{2}Vi completed the task in just ~22 sec. Thus, despite the hardware limitations, ProS^{2}Vi delivers efficient results, whereas the web-based tools could be offline occasionally due to server technical challenges.

In this comparison, it is noteworthy to elaborate on the reasons why ProS^{2}Vi can take more time to complete a task than POLYVIEW-2D and STRIDE do for specific PDB files. The POLYVIEW-2D and STRIDE simplify secondary structure assignments into just four categories, α-helices and other helices with only two viewing options, β-strands or bridges, and coils. In contrast, ProS^{2}Vi uses a more detailed classification scheme, visualizing eight distinct structure categories consisting of α-helices, 3–10 helices, pi helices, isolated beta-bridge residues, strands, turns, bends, and also unsolved structures according to the detailed DSSP output. Additionally, ProS^{2}Vi assigns an index to each folding element, enabling precise structural tracking and analysis across the protein structure visualization. This broad classification provides a rich and comprehensive visualization for researchers with precise structural insights based on the DSSP algorithm [6].

The tool features a contemporary design with customizable icon colors and the ability to export visualizations as scalable PDFs. Modern elements of the GUI ensure that users can intuitively navigate the application, customize their visualizations according to their preferences, and produce high-quality outputs suitable for presentations and publications. Furthermore, the ability to directly download a protein structure from RCSB using the PDB code adds to the accessibility and ease of use. Even though ProS^{2}Vi can connect to RCSB to download protein structures, any uploaded protein structure files are stored and used locally, keeping security a top priority.

This modern interface not only improves the aesthetic appeal but also enhances functionality, making the tool more approachable and easier to use compared to existing solutions such as POLYVIEW-2D [9] and STRIDE [8]. Neither of these tools accepts a "mmCIF" file format, which is essential because some of the larger protein structures are only supported by the "mmCIF" file format. Furthermore, neither of the tools provides the option to change or modify the color of structural icons, and additionally, STRIDE is unable to offer the option to save or download a high-quality image of the visualization easily. Most importantly, none can be used locally to generate a protein's secondary structure diagrams. Our tool also automatically saves the generated image and can also create a PDF document. Additionally, ProS^{2}Vi's [13] visualization also includes the protein structure's title and source scientific name with the help of the RCSB application programming interface (API) [46], UniProt ID for each chain using the Proteins API [47], and a legend (i.e., key) (Figure S1).

ProS^{2}Vi is designed to integrate into broader bioinformatics workflows seamlessly. It supports both "PDB" and "mmCIF," which are widely used in bioinformatics pipelines. Furthermore, using the command-line interface (CLI) version of ProS^{2}Vi, users can automate the visualization process and integrate it into scripted workflows without any usage caps; this allows ProS^{2}Vi to be easily combined with other bioinformatics tools like annotation pipelines, for high-throughput analysis.

Another key advantage of ProS^{2}Vi is that it is open source, allowing researchers and developers to collaborate, customize, and extend its functionality. This open development model not only encourages improvements but also facilitates integration with other tools, as users can modify the codebase to suit their specific research needs. ProS^{2}Vi not only fits well into existing bioinformatics pipelines but also empowers the research community to enhance its capabilities through collaboration and automation.

For the technical users, the output can be further customized by adding changes to the accompanying Cascading Style Sheets (CSS) according to user preference. This file is named "output_styles.css" and is located inside the "templates" folder. The value of the "font-family" property inside the "body" selector can be changed to any other font supported by the web browser if a user prefers to change the font style. Font size can also be easily changed by modifying the values of the "font size" appropriately for different parts of the output such as "title," "legend," etc.

4. Conclusion

ProS^{2}Vi [13], a Python-based visualization tool, offers a secure, advanced, and user-friendly solution for visualizing protein secondary structures. Its local operation provides enhanced security and control, eliminating the risks associated with online platforms, ensuring continuous availability without downtime, and enabling compatibility with automated workflows through its command-line interface. As an open-source tool, ProS^{2}Vi allows collaboration and improvements from the research community, fostering innovation and expanding its capabilities over time. The modern interface and broad file format compatibility make it a versatile tool for researchers and educators alike. An intuitive graphical user interface allows users to easily navigate the tool and perform complex analyses without requiring extensive coding skills or computational background, enabling a broader range of users, from students to researchers, to benefit from this tool. The GUI is designed to streamline workflow, allowing users to load protein structures, process them through the DSSP algorithm, and visualize the results with minimal steps.

Additionally, ProS^{2}Vi's ability to export visualizations as scalable PDFs is a significant advantage, allowing users to produce high-quality images. This feature ensures that the visualizations can be scaled up for presentations and publications without losing image quality. ProS^{2}Vi stands out for generating detailed 2D folding visualizations with eight structure categories and assigning an index to each fold, providing a comprehensive view of secondary structures compared to other tools. Despite relying on DSSP for structure assignments, which affects speed improvements, ProS^{2}Vi delivers efficient performance, with notable

speed advantages for some PDBs, even on modest hardware. Its memory management is efficient, given the complexity of its visual outputs. Overall, ProS^{2}Vi's combination of security, customizability, detailed visualization, and open-source collaboration makes it an asset in structural biology and bioinformatics research.

The software is readily available at https://github.com/Alisaraie-Group/ProS2Vi (DOI:10.5281/zenodo.12554830).

Funding sources

This work was supported by funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), Discovery Grant (No. 212654), awarded to LA.

CRediT authorship contribution statement

Alisaraie Laleh: Writing – review & editing, Validation, Supervision, Project administration, Investigation, Funding acquisition, Conceptualization. **Qasim Luckman:** Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. The manuscript was written with contributions from all authors, and all authors have approved the final version.

Declaration of Competing Interest

The authors have no conflict of interest to declare.

Appendix A. Supporting information

Additional visualization figures comparing different Protein secondary structure visualization software (DOC). Supplementary data associated with this article can be found in the online version at doi:10.1016/j.csbj.2025.02.038.

Data availability

The ProS^{2}Vi [13] source code is freely available at https://github.com/Alisaraie-Group/ProS2Vi (DOI:10.5281/zenodo.12554830) under an open-source Apache 2.0 license.

References

[1] Brandt, G.S.. Secondary structure. In: Wells, R.D., et al., editors, Molecular life sciences: an encyclopedic reference. New York, NY: Springer New York; 2018. p. 1089-97.
[2] O'Donoghue SI. Grand challenges in bioinformatics data visualization. Front Bioinform 2021;1:669186.
[3] Koch O. Use of secondary structure element information in drug design: polypharmacology and conserved motifs in protein-ligand binding and protein-protein interfaces. Future Med Chem 2011;3(6):699–708.
[4] Frishman D, Argos P. Knowledge-based protein secondary structure assignment. Proteins 1995;23(4):566–79.
[5] Martin J, et al. Protein secondary structure assignment revisited: a detailed analysis of different assignment methods. BMC Struct Biol 2005;5(1):17.
[6] Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 1983;22(12):2577–637.
[7] Zhang Y, Sagui C. Secondary structure assignment for conformationally irregular peptides: comparison between DSSP, STRIDE and KAKSI. J Mol Graph Model 2015;55:72–84.
[8] Heinig, M. and Frishman, D. STRIDE: a web server for secondary structure assignment from known atomic coordinates of proteins. Nucleic Acids Res 2004; 32 (Web Server issue):W500-W502.
[9] Porollo AA, Adamczak R, Meller J. POLYVIEW: a flexible visualization tool for structural and functional annotations of proteins. Bioinforma (Oxf, Engl) 2004;20 (15):2460–2.
[10] Gu Q, Liu P. Denial of service attacks. In: Handbook of computer networks. John Wiley and Sons; 2012. p. 454–68.
[11] Nawaz NA, et al. A comprehensive review of security threats and solutions for the online social networks industry. PeerJ Comput Sci 2023;9:e1143.
[12] Amo-Filva D, et al. Local technology to enhance data privacy and security in educational technology. Int J Interact Multimed Artif Intell 2021;7:262.
[13] Qasim, M.L.Alisaraie, L. 2024. ProS2Vi: a Python tool for visualizing proteins secondary structure. Available from: ⟨https://github.com/Alisaraie-Group/ProS2Vi/⟩.
[14] Roelofs G, Koman RPNG. The definitive guide. O'Reilly & Associates, Inc; 1999.
[15] Hudson G, et al. JPEG at 25: still going strong. IEEE Multimed 2017;24(2):96–103.
[16] Grech V. The portable document format - PDF. Images Paediatr Cardiol 2002;4(2):1–3.
[17] Consortium TU. UniProt: the Universal Protein Knowledgebase in 2023. Nucleic Acids Res 2022;51(D1):523–D531 (D).
[18] Burley SK, et al. Protein Data Bank (PDB): the single global macromolecular structure archive. Methods Mol Biol (Clifton, N J) 2017;1607:627–41.
[19] Schrödinger, La.W.D. 2020. PyMOL. Release 2.4.0. Available from: ⟨https://www.pymol.org/⟩, ⟨http://www.pymol.org/pymol⟩. [Accessed 20 May 2020].
[20] Jmol-Development-Team. 2016. Jmol. Release 14.6.4. Available from: ⟨http://jmol.sourceforge.net/⟩. [Accessed 7 October 2016].
[21] Pettersen EF, et al. UCSF Chimera–a visualization system for exploratory research and analysis. J Comput Chem 2004;25(13):1605–12.
[22] Westbrook JD, et al. PDBx/mmCIF ecosystem: foundational semantic tools for structural biology. J Mol Biol 2022;434(11):167599.
[23] Cock PJ, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinforma (Oxf, Engl) 2009;25(11):1422–3.
[24] Project, T.D. 1998. Advanced packaging tool (APT). Available from: ⟨https://ubuntu.com/server/docs/package-management⟩.
[25] Ronacher, A. 2008. Jinja. Available from: ⟨https://jinja.palletsprojects.com/en/3.1.x/⟩.
[26] Truelsen, J. and Kulkarni, A. 2009. wkhtmltopdf. Available from: ⟨https://github.com/wkhtmltopdf/wkhtmltopdf⟩.
[27] JazzCore 2013. Python-PDFKit: HTML to PDF wrapper. Available from: ⟨https://github.com/JazzCore/python-pdfkit⟩.
[28] Jarrekk. IMGKit: Python library of HTML to IMG wrapper. 2017 ⟨https://github.com/jarrekk/imgkit⟩.
[29] Oakley AJ, et al. The structures of human glutathione transferase P1-1 in complex with glutathione and various inhibitors at high resolution. J Mol Biol 1997;274(1):84–100.
[30] Bhatt H, et al. Structure of an unfolding intermediate of an RRM domain of ETR-3 reveals its native-like fold. Biophys J 2020;118(2):352–65.
[31] Vemu A, et al. Structure and dynamics of single-isoform recombinant neuronal human tubulin. J Biol Chem 2016;291(25):12907–15.
[32] Van Der Spoel D, et al. GROMACS: fast, flexible, and free. J Comput Chem 2005;26 (16):1701–18.
[33] Microsoft. 2016. Windows subsystem for Linux. Available from: ⟨https://learn.microsoft.com/en-us/windows/wsl/install⟩.
[34] Ronacher A. Flask web development: developing web applications with Python 2010. ⟨https://flask.palletsprojects.com/en/3.0.x/⟩.
[35] Belval, E. 2017. pdf2image. Available from: ⟨https://github.com/Belval/pdf2image⟩.
[36] Kozea. 2011. CairoSVG. Available from: ⟨https://cairosvg.org/⟩.
[37] Clark, J.A. 2010. Pillow. Available from: ⟨https://pillow.readthedocs.io/en/stable/⟩.
[38] Prewitt, N., Cordasco, I. and Larson, S.M. 2011. Requests: HTTP for humans™. Available from: ⟨https://requests.readthedocs.io/en/latest⟩.
[39] Burley SK, et al. RCSB Protein Data Bank (RCSB.org): delivery of experimentally-determined PDB structures alongside one million computed structure models of proteins from artificial intelligence/machine learning. Nucleic Acids Res 2022;51 (D1):488–D508 (D).
[40] Sendker FL, et al. Emergence of fractal geometries in the evolution of a metabolic enzyme. Nature 2024;628(8009):894–900.
[41] Cottevieille M, et al. The subnanometer resolution structure of the glutamate synthase 1.2-MDa hexamer by cryoelectron microscopy and its oligomerization behavior in solution: functional implications. J Biol Chem 2008;283(13):8237–49.
[42] Rienstra CM, et al. De novo determination of peptide structure with solid-state magic-angle spinning NMR spectroscopy. Proc Natl Acad Sci USA 2002;99(16):10260–5.
[43] Sibanda BL, Chirgadze DY, Blundell TL. Crystal structure of DNA-PKcs reveals a large open-ring cradle comprised of HEAT repeats. Nature 2010;463(7277):118–21.
[44] Aher A, et al. Structure of the γ-tubulin ring complex-capped microtubule. Nat Struct Mol Biol 2024;31(7):1124–33.
[45] Sapin S, Ayoub G, Anglade L. WeasyPrint 2011. ⟨https://weasyprint.org/⟩.
[46] Rose Y, et al. RCSB protein data bank: architectural advances towards integrated searching and efficient access to macromolecular structure data from the PDB archive. J Mol Biol 2021;433(11):166704.
[47] Nightingale A, et al. The Proteins API: accessing key integrated protein and genome information. Nucleic Acids Res 2017;45(W1):W539–44.