Complete User Documentation for the Docking Application

1. Overview

This web-based application provides an automated pipeline for molecular docking with AutoDock Vina (v1.2.5) and optionally the P2RANK tool (for automatic binding-pocket prediction in "SMART DOCKING"). The environment runs inside a Streamlit app (dock_GUI.py or dock_GUI_container.py) that interacts with two main back-end Python scripts for docking:

1. **init_docking.py** (the “SMART DOCKING” script):
   - For automatically downloading PDB structures (or mmCIF if PDB unavailable) from the RCSB database, cleaning them, optionally performing an automatic search for pockets via P2RANK, and then performing docking with specified parameters.
2. **init_docking2.py** (the “MANUAL DOCKING” script):
   - For manually prepared receptors in .pdbqt format along with matching parameter CSV files. This script does not rely on P2RANK or receptor download/cleanup steps.

The user interacts entirely through the Streamlit interface, which handles:

- **Authentication** (login, registering new users)
- **Project creation** (naming your docking run)
- **Uploading** ligands, receptors, parameter files
- **Launching** docking on a cluster or local SLURM queue (the code typically calls sbatch, but can be adapted)
- **Queue** monitoring & job cancellation
- **Viewing** results in an interactive HTML format (optionally served from a local HTTP server for easy 3D/HTML viewing)
- **Downloading** results in bulk .zip archives
- **Deleting** old projects

2. User Login & Registration

When you visit the application URL (e.g., if running on a server or on localhost), you will see:

1. **Login Screen**: Enter your username and password.
   - If valid credentials exist, you are granted access.
   - If not, you may see an error message: “Wrong username or password.”
   - If the system finds no password file, you must register a user first.
2. **Registration** (Add New User):
   - You can register a new username (in lower-case letters) and a password.
   - The password file is stored server-side.

Once logged in, you are brought to the main menu.

3. Main Menu

After authentication, the menu displays the following options:

1. **SMART DOCKING**
2. **MANUAL DOCKING**
3. **QUEUE**
4. **SHOW RESULTS**
5. **DOWNLOAD RESULTS**
6. **DELETE RESULTS**
7. **PyMOL Installation GUIDE**
8. **LOG OUT**

Below is a detailed explanation of each module.

3A. SMART DOCKING

This is the "automatic" pipeline that downloads PDB structures, uses P2RANK for pocket detection, and sets up docking.

1. **Project Setup**
   - Provide a unique project name. The interface creates a folder named username_projectName and copies a template folder inside it.
   - If a project with that name already exists, you may proceed or rename.
2. **Input PDB Codes or Flexible Docking**
   - You can enter multiple PDB IDs (comma-separated) or upload a CSV with "PDB_code,Chain_ID" lines.
   - The system downloads each PDB or mmCIF from the RCSB, cleans it, and shows you available chains.
   - Optionally, if you want flexible docking, you may provide rigid.pdbqt and flex.pdbqt in the relevant step.
3. **Ligand Upload**
   - Choose a single .mol2 or .sdf file containing one or multiple ligands.
4. **Docking Parameters**
   - You can adjust exhaustiveness, number of modes, energy range, etc., and whether to keep original chain IDs or perform a small "relaxation."
   - The interface can import or export these parameters to a docking_parameters.csv.
5. **Summary**

    - You see the final summary: Receptors, chain IDs, the ligand file, the parameters, etc.

6. **Start Docking**
    - Press “Start Docking” to submit the job.
    - The interface creates a Slurm batch script (start_docking.sh) that calls init_docking.py with the chosen arguments.
    - The job is queued, and you can check it in the QUEUE module.

Result: When the docking completes, the back-end script writes:

- Output text logs
- .pdbqt results for each ligand
- Visualization images
- An HTML summary with advanced pocket references (thanks to P2RANK)
- A CSV summary of energies

These can be viewed in the “SHOW RESULTS” or downloaded in “DOWNLOAD RESULTS.”

3B. MANUAL DOCKING

This module is for advanced or manual scenarios where you have already prepared receptor .pdbqt files (outside of this pipeline), along with a parameter .csv for each receptor specifying the grid location and size. The script init_docking2.py does **not** do any receptor cleaning or P2RANK runs.

Workflow:

1. **Project Setup**
    - Again, provide a project name. The system creates or re-uses your project folder (with a “template2” if needed).
2. **Files Upload**
    - **Receptor Files**: Place all your .pdbqt receptors in the “receptors” folder via the Streamlit interface.
    - **Parameter Files**: For each receptor named, e.g., XYZ.pdbqt, you must provide a CSV named XYZ.csv with lines such as:
    - parameter,value
    - exhaust,40
    - energy_range,1
    - num_modes,5
    - seed,1988
    - grid_center_x,107

  - grid_center_y,114
  - grid_center_z,143
  - grid_size_x,17
  - grid_size_y,20
  - grid_size_z,35

The script reads these to set the docking grid and Vina arguments.

  - **Ligand File**: One .sdf or .mol2 containing all ligands.

3. **Summary**
   - The interface lists how many .pdbqt files, how many parameter .csvs, and the chosen ligand file.
4. **Start Docking**
   - When you confirm, the system writes a batch script calling init_docking2.py.
   - That script converts your ligand(s) to .pdbqt, reads each receptor's CSV, and runs AutoDock Vina in a loop for each ligand–receptor pair.
   - It stores images, a text file of all docking poses, a final CSV with (name, affinity, smiles), and an HTML summary.

The "MANUAL DOCKING" approach is best if you have your own grid definitions, or multiple custom receptors pre-processed outside the pipeline.

---

4. QUEUE (SLURM Jobs)

This module displays the Slurm job queue by calling squeue. You can:

- **REFRESH** to re-run squeue and see an updated table of jobs.
- **Cancel Job**: Provide a "JOB_ID" and if the job name belongs to you (matching username_jobName), it will let you cancel it (scancel JOB_ID).

---

5. SHOW RESULTS

In "SHOW RESULTS," you can select an existing project, choose a receptor, and if an interactive .html file is present (with advanced 3D or 2D visual elements), you can open it. The system can serve those static files via the local HTTP server so that you can see embedded structures or interactive JS-based viewers.

If the script generated a CSV file in that receptor's folder, you can also download it individually (the table of final docking poses).

---

6. DOWNLOAD RESULTS

You can select one or more projects, then press “Download selected projects” to receive a ZIP containing all files. This includes:

- .pdbqt final receptor or ligand poses
- .log or console output
- .txt or .csv summaries
- Images, PyMOL session files, and HTML

---

7. DELETE RESULTS

You may highlight one or multiple projects and permanently remove them from the server (the pipeline calls shutil.rmtree). Be sure you have downloaded any essential data first!

---

8. PyMOL Installation Guide

A small page describing how to install and run open-source PyMOL (for local offline visualization) on Windows or Linux is included.

---

9. Implementation Details & Notes

1. **Authentication & User Management**:
   - The credentials are stored in a CSV: passwords.pw in the results/passwords directory.
   - Password hashing uses bcrypt.
2. **Project Folders**:
   - Named username_projectName, typically located under results/.
   - For “SMART DOCKING,” you might see subfolders: receptors, ligands, and a template that includes scripts or placeholders.
   - For “MANUAL DOCKING,” you might see a similar structure but referencing a different template2.
3. **Running the Scripts**:
   - Each step sets up environment variables or calls python3 init_docking.py or python3 init_docking2.py with your chosen arguments.
   - If using a cluster, each final step writes out a Slurm script and calls sbatch start_docking.sh (or start_docking_manual.sh).
4. **Queue**:
   - The queue module runs a command like squeue -r -o '%i,%j,%T,%M,%V' --noheader and parses the text.
   - Cancelling a job checks if the job’s name starts with the user’s name.

5. **Data Visualization**:
    - The scripts generate .png or .svg images for 2D ligand depiction, 3D docking images, and optional .pse PyMOL session files.
    - The advanced HTML can include embedded images, clickable links, and partial local JS or CSS styling.
6. **Important**: Ensure OBABEL_PATH, VINA_PATH, P2RANK_PATH are correctly set in your environment, or you'll see "File not found" errors.

---

10. Summary

1. **Log in** to the Streamlit app.
2. **Pick a module** (SMART or MANUAL).
3. **Create or select** a project, upload data, set parameters.
4. **Launch** the docking job.
5. **Monitor** the queue.
6. **View** or **download** results.

Should any error occur (e.g., job script failing, or the server lacking installed dependencies), check the console logs or the .log files in each project's folder. For advanced usage or debugging, refer to the included Python scripts:

- init_docking.py (auto)
- init_docking2.py (manual)

and read their docstrings in detail to see exactly how the docking is performed.

End of Document.