Project steps:

Your code is a good starting point, and it already covers the basic functionality of listing, adding, and deleting entries from the phonebook. To meet the requirements of the assignment, you need to incorporate input validation and handle error cases appropriately. Additionally, you should implement the audit log, create Docker containers, and write security-focused unit tests.

Here are some steps to enhance your code:

### 1. Input Validation:

1.1 \*\*Implement Regular Expressions:\*\*

- Create regular expressions for validating the `full\_name` and `phone\_number` in the `Person` class. Ensure they comply with the assignment's rules for acceptable inputs.

1.2 \*\*Validate Inputs:\*\*

- Add code to validate the `full\_name` and `phone\_number` using the defined regular expressions.

- If the validation fails, return an appropriate HTTP 400 response with an error message.

### 2. Error Handling:

2.1 \*\*Handle Non-existent Entities:\*\*

- For the delete operations (`delete\_by\_name` and `delete\_by\_number`), check if the person exists before attempting to delete. If not, return an HTTP 404 response.

### 3. Audit Log:

3.1 \*\*Implement Audit Logging:\*\*

- Create a function to log timestamped entries for add, remove, and list operations.

- Include the name of the person added/removed in the log.

### 4. Dockerization:

4.1 \*\*Create Docker Containers:\*\*

- Write Dockerfiles to package your application and its dependencies into containers.

- Ensure the containers include all necessary dependencies for your technology stack.

### 5. Security-focused Unit Tests:

5.1 \*\*Write Automated Tests:\*\*

- Implement automated unit tests to cover various scenarios, including valid and invalid inputs for names and phone numbers.

- Use a testing framework like `pytest` to organize and run your tests.

### 6. Documentation:

6.1 \*\*Write Instructions:\*\*

- Provide clear and concise instructions on how to build, run, and test your application using Docker containers.

- Include information about any configuration files or environment variables required.

### 7. Report:

7.1 \*\*Prepare a Report:\*\*

- Describe how your code works, detailing the implemented features.

- Document the design of your regular expressions and any assumptions made.

- Discuss the pros and cons of your approach.

### 8. Bonus (Optional):

8.1 \*\*Database Integration:\*\*

- If you choose to attempt the bonus, switch to using a SQL database engine (e.g., SQLite).

- Use parameterized queries to prevent SQL injection vulnerabilities.

### 9. Security Tests:

9.1 \*\*Implement Security Tests:\*\*

- Write security-focused unit tests to verify both good and bad inputs, as specified in the assignment.

### 10. Final Testing:

10.1 \*\*Test Your Application:\*\*

- Test your application thoroughly using the provided sample inputs and additional test cases.

### 11. Submission:

11.1 \*\*Prepare Submission:\*\*

- Organize your code, Dockerfiles, and instructions for submission.

- Ensure all dependencies are included in the Docker containers.

By following these steps, you'll be able to address the assignment's requirements and provide a well-documented, secure, and functional solution. If you have any specific questions or encounter challenges during implementation, feel free to ask!