PROJECT DESCRIPTION

The task of the final project is to develop a web application that supports functionality according to the task variant.

Implementation requirements:

1. Create classes based on the entities of the subject domain.
2. Classes and methods should have names that reflect their functionality and should be organized into packages.
3. Code formatting should follow the Java Code Convention.
4. Store information about the subject domain in a relational database (MySQL or PostgreSQL is recommended as the DBMS).
5. Use JDBC API for data access with the utilization of a ready-made or self-developed connection pool.

The use of ORM frameworks is not allowed.

---

6. The application should support working with Cyrillic (be multilingual), including storing information in the database:
    a. There should be an option to switch the language of the interface.
    b. Input, output, and storage of information (in the database) should be supported for different languages.
    c. Choose at least two languages: one based on Cyrillic (Ukrainian or Russian) and another based on Latin (English).
7. The application architecture should follow the MVC pattern.

The use of MVC frameworks is not allowed.

---

8. When implementing the business logic, it is necessary to utilize design patterns such as Command, Strategy, Factory, Builder, Singleton, Front Controller, Observer, Adapter, etc.

The use of design patterns should be justified.

---

9. Implement the functionality outlined in the task using servlets and JSP.
10. Use Apache Tomcat as the servlet container.
11. Apply JSTL tags and develop custom tags (minimum: one custom tag library and one tag file) in JSP pages.
12. Implement protection against duplicate form submission on page refresh (implement PRG).
13. Use sessions, filters, and listeners during development.

14. The application should include authentication and authorization, differentiating user access rights to program components. Password encryption is encouraged.
15. Implement an event log using the log4j library.
16. The code should include documentation comments (for all top-level classes, nontrivial methods, and constructors).
17. The application should be covered by unit tests (minimum code coverage of 40%). Integration tests are encouraged.
18. Implement a pagination mechanism for data pages.
19. All input fields should have data validation.
20. The application should handle errors and various exceptional situations properly (the end-user should not see a stack trace on the client side).
21. Self-extension of the task specification regarding functionality is encouraged! (adding CAPTCHA, generating reports in different formats, etc.)
22. The use of HTML, CSS, JS frameworks for the user interface (Bootstrap, Materialize, etc.) is encouraged!

Three days before the start of project defenses (interviews), prepare a separate file with the database schema and provide a link to the project repository.
