ДОДАТОК А

Керівництво користувача

ЗМІСТ

ВСТУП

Документ Керівництво користувача призначено для використання працівниками підприємства. Він надає інформацію про функціонал системи, правила роботи з нею та рекомендації щодо дій у надзвичайних ситуаціях.

Інтерфейс розробленої системи створено простим та інтуїтивно зрозумілим користувачам будь-якого рівня. Доступ до нього мають лише ті працівники, які були раніше зареєстровані адміністратором (при прийомі на роботу).

Новим користувачам системи настійно рекомендується ознайомитися з документом перед початком роботи в системі для розуміння функціоналу системи та забезпечення ефективної роботи з нею.

1 ЗАГАЛЬНИЙ ОПИС СИСТЕМИ ТА ЇЇ ФУНКЦІОНАЛУ

Система має три ролі користувачів. Проєктувальник – людина, яка планує графік роботи підприємства, створює робочі програми для машин та перевіряє їх правильність. Диспетчер – людина, яка відповідає за моніторинг робочих процесів підприємства, фіксування аномалій та неполадок і вчасне повідомлення про них ремонтну бригаду. Бригадир ремонтної бригади – людина, яка керує ремонтом неполадок, може змінювати статуси машин в залежності від того, чи відправлені вони на склад через несправність.

Застосунок реалізовує такий функціонал:

– аутентифікація: функція, яка дозволяє користувачеві зайти в систему та отримати потрібний для нього інтерфейс;

– перегляд замовлень, що надійшли: проєктувальник може переглядати всі замовлення від клієнта, які надійшли в систему;

– створення робочих програм: проєктувальник може створювати робочі програми, використовуючи інформацію з замовлень;

– перегляд програм та імітація їх виконання: проєктувальник може переглядати створені ним програми та, шляхом переходу на іншу вкладку, передивлятися рівень завантаженості машин на поточний день у графічному вигляді. У разі перевантаження графік буде червоного кольору, тому проєктувальник повинен буде редагувати програми, щоб уникнути даного явища;

– перегляд створених програм для одного замовлення: проєктувальник може переглядати створені програми, які згруповані по номеру замовлення, а також переглядати персональні завдання для кожної машини по кожній програмі, в яких наведено детальну інформацію щодо виконання процесу;

– перегляд денного рапорту: проєктувальник та диспетчер можуть переглядати рапорт, який формується за день роботи підприємства, та в якому детально зазначено, які програми виконалися та чи був проведений ремонт.

– перегляд програм, заплаанованих на поточний день: диспетчер при вході в систему бачить список у таблиному вигляді з програмами, які були заплановані проєктувальником на цей день. Також є можливість переглядати персональні завдання на кожну машину;

– моніторинг показників підприємства: диспетчер при запуску робочих процесів може переглядати як детальні показники машин, що змінюються, так і загальні, однакові для всіх;

– виклик ремонтної бригади: при надзвичайних ситуаціях диспетчер може зупинити виробничі процеси та створити рапорт на ремонт для бригади;

– перегляд статусів рапорту: після відправки рапорта його статус показується як Відправлено. При зміні даних рапорту бригадиром статус стане Виконано. Диспетчер може відслідковувати зміну статусів на відповідній вкладці;

– редагування рапорту: бригадир, який отримує рапорт диспетчера, повинен заповнити деякі поля після завершення ремонту.

– зміна статусів машин: при заміні непрацюючої машини на підприємстві бригадир може замінити її на іншу у відповідній вкладці сайту для відображення змін у системі.

2 ПІДГОТОВКА ДО РОБОТИ З СИСТЕМОЮ

Перед початком роботи з системою потрібно впевнитися в тому, що системні вимоги для її запуску було виконано:

Серверна частина:

– підключення до локального хосту за допомогою MAMP або подібних додатків;

– вебдодаток для взаємодії з базою даних PhpMyAdmin;

– інтегрована середа розробки програмного забезпечення з підтримкою мови Java з встановленим на ньому JDK 20 і вище.

– фреймворк для збірки проєктів Apache Maven 3.8.1 і вище;

– фреймворк для роботи з проєктом Spring Boot 3.0 і вище;

– ORM: Hibernate 5.0 і вище;

Клієнтська частина:

– шаблонизатор Thymeleaf;

– фреймворк Bootstrap;

– веббраузер Mozilla Firefox, Google Chrome, Opera та інші.

Також необхідно виконання таких вимог:

– пристрій користувача підключено до мережі Інтернет;

– користувач ознайомлений у повній мірі з Керівництвом користувача та володіє знаннями щодо взаємодії з системою.

Якщо наведені вище вимоги виконані, користувач повинен відкрити браузер та прописати у рядку пошуку http://localhost:8080 і натиснути клавішу Enter, після чого йому буде доступна сторінка авторизації сайту.

3 ОПИС ОПЕРАЦІЙ

3.1 Авторизація користувача

Кожен користувач системи має створений для нього обліковий запис, тому для входу в нього необхідно авторизуватися в системі. Для цього на сторінці входу користувачеві необхідно ввести свій логін та пароль і натиснути кнопку Login. Сторінка авторизації користувача наведена на рис. А.1.

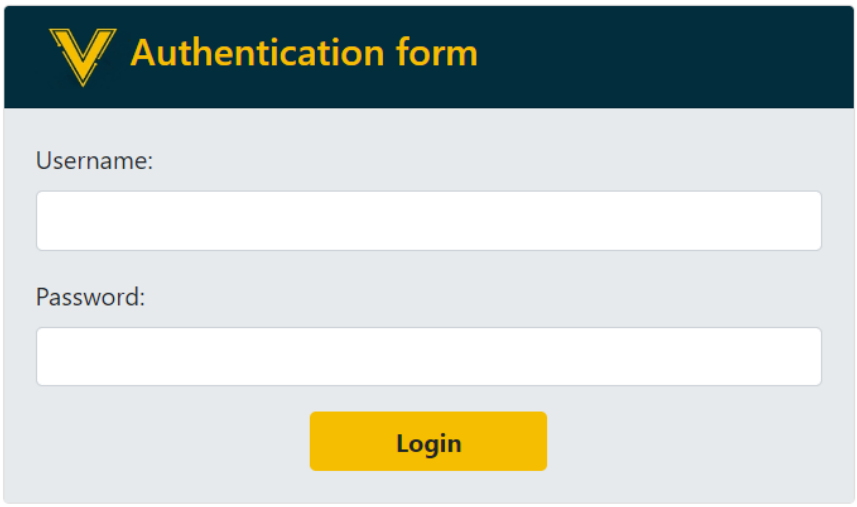


Рисунок А.1 – Сторінка авторизації користувача

Якщо користувач ввів неправильні дані для входу, система сповістить його про це повідомленням про помилку введення інформації. В даному випадку необхідно ввести коректні дані та знову натиснути кнопку Login. На рисунку А.2 зображено сторінку авторизації з повідомленням про некоректне введення даних.

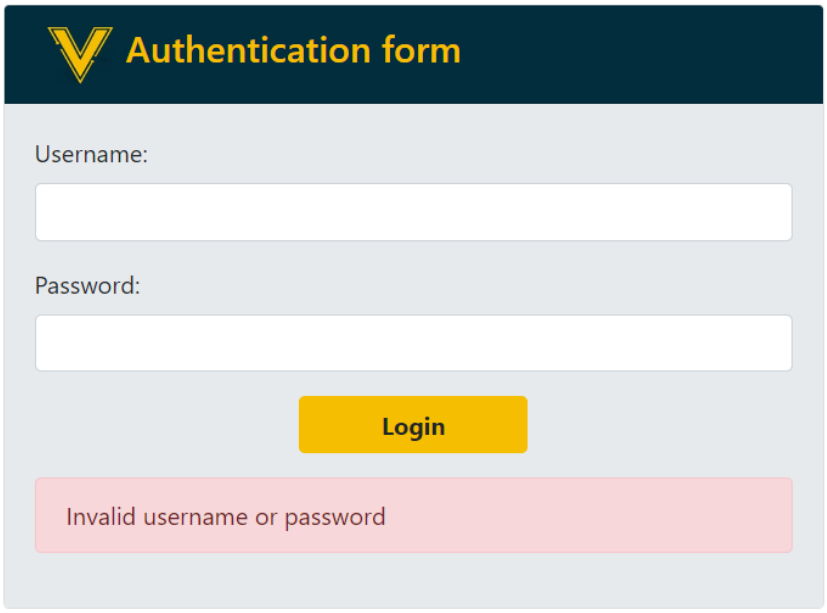
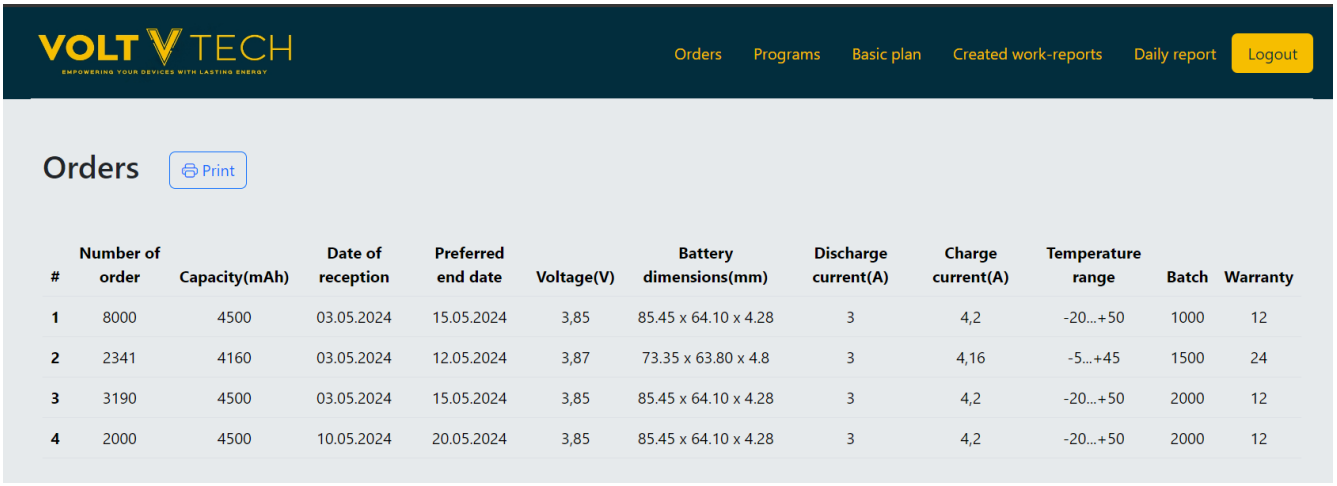


Рисунок А.2 – Сторінка з помилкою при введенні даних

При успішній авторизації користувач потрапляє до того інтерфейсу, який був наданий системою в залежності від його ролі. Якщо користувач має роль Проєктувальник, то інтерфейс його головної сторінки буде таким (рис. А.3):



#	Number of order	Capacity(mAh)	Date of reception	Preferred end date	Voltage(V)	Battery dimensions(mm)	Discharge current(A)	Charge current(A)	Temperature range	Batch	Warranty
1	8000	4500	03.05.2024	15.05.2024	3,85	85.45 x 64.10 x 4.28	3	4,2	-20...+50	1000	12
2	2341	4160	03.05.2024	12.05.2024	3,87	73.35 x 63.80 x 4.8	3	4,16	-5...+45	1500	24
3	3190	4500	03.05.2024	15.05.2024	3,85	85.45 x 64.10 x 4.28	3	4,2	-20...+50	2000	12
4	2000	4500	10.05.2024	20.05.2024	3,85	85.45 x 64.10 x 4.28	3	4,2	-20...+50	2000	12

Рисунок А.3 – Головна сторінка інтерфейсу користувача Проєктувальник

Якщо користувач авторизувався від імені диспетчера, то головна сторінка його інтерфейсу буде такою (рис. А.4).

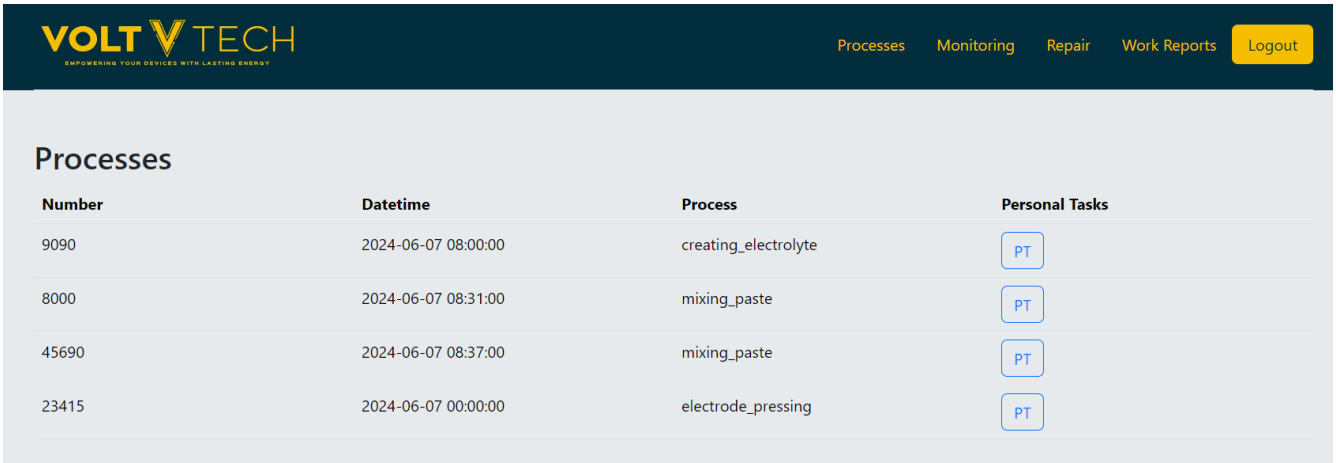


Рисунок А.4 – Головна сторінка інтерфейсу користувача Диспетчер

Якщо користувач авторизувався від імені бригадира ремонтної бригади, то головна сторінка його інтерфейсу буде такою (рис. А.5):

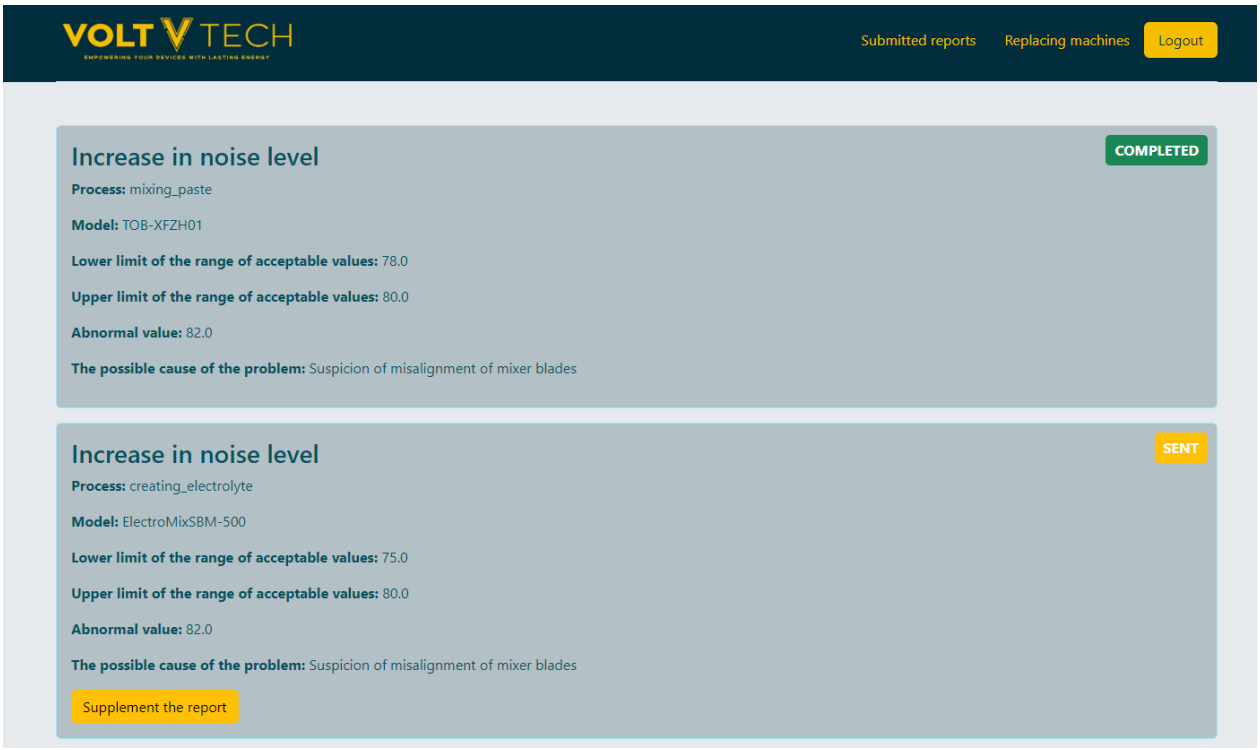


Рисунок А.5 – Головна сторінка користувача Бригадир ремонтної бригади

Для виходу з системи користувач має натиснути кнопку Logout, яка знаходиться в шапці сайту кожного інтерфейсу, і він буде перенаправлений на сторінку аутентифікації.

3.2 Опис функціоналу для користувача Проєктувальник

3.2.1 Перегляд замовлень, що надійшли

На головній сторінці додатку при вході з боку проєктувальника одразу відкриваються замовлення клієнтів, які надійшли в систему (рис. А.3). Їх можна роздруковувати кліком на кнопку Print.

3.2.2 Створення робочих програм

При переході на вкладку Programs користувач бачить поля для введення даних про створюванні програми:

– Enter begin date: поле для введення дати та часу, коли дана програма повинна почати виконуватися машинами підприємства;

– Enter id-number: номер замовлення, який проєктувальник бере з таблиці замовлень;

– Enter batch identifier: номер партії, якщо одна робоча програма ділиться на декілька;

– Select title: назва робочої програми, обирається з випадаючого списку, залежить від машини, якій ця програма належить;

– Enter number of batch: кількість деталей в партії, початкове значення береться з таблиці замовлень;

– Select capacity: значення ємності акумулятора, обирається з випадаючого списку згідноз даними з таблиці замовлень;

– Select process: процес на виробництві, для якого створюється програма, обирається з випадаючого списку;

– Select machine: тип машини, яка повинна виконувати вказаний раніше процес, обирається з випадаючого списку;

– Processing time: значення, яке підставляється автоматично при виборі типу машини, означає час, за який машина створить один акумулятор, або кількість матеріалу, необхідну для одного акумулятора (для міксерів);

Після створення програми користувач натискає на кнопку Add work-program та перенаправляється на вкладку Basic plan, де відображена створена програма. Вид сторінки Program наведено на рисунку А.6.

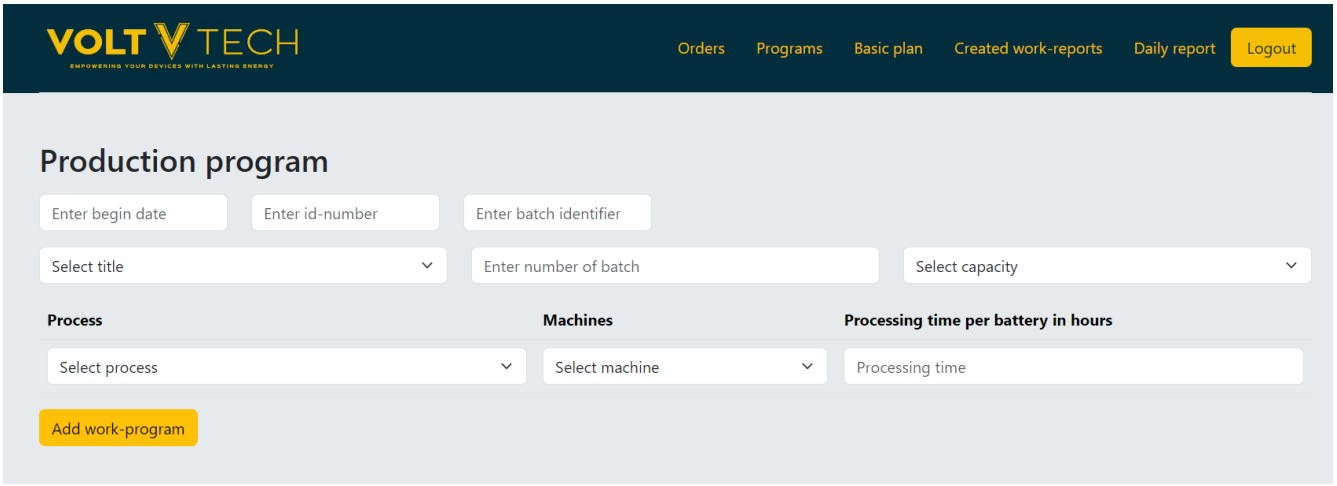


Рисунок А.6 – Сторінка створення робочих програм

3.2.3 Перегляд створених програм та імітація їх виконання

На вкладці Basic plan користувачеві доступні усі створені ним раніше програми, які оформлені у блоки за назвою, вміст блоків можна відкривати та ховати, натискаючи на кнопку Hide/Show. Під кожною програмою міститься дві кнопки, Update (перенаправляє користувача на сторінку редагування програми) та Delete (дозволяє видалити дану робочу програму).

На даній сторінці присутні внутрішні вкладки. Та, на якій відображені всі програми, має назву Basic plan of programs, інша має назву Diagram. На другій внутрішній вкладці відображається імітація графіку роботи даної машини по днях,

за його допомогою можна відслідковувати її завантаженість. На осі ординат знаходиться дата виконання програми, на осі абсцис – час, який займає виконання цих програм.

Для кожного блоку машин графік різний, тому доступ до нього надається за допомогою натискання на необхідний блок (він підсвічується іншим кольором) і переходу на вкладку Diagram. Якщо у цей день машина перевантажена робочими процесами, то колір графіку цього дня буде червоним. Якщо завантаження машини недостатнє – графік буде жовтого кольору. Якщо завантаження відповідає необхідному проміжку оптимальної завантаженості – графік буде зеленого кольору. Вигляд вкладки Basic plan of programs з блоками відображено на рисунку А.7.

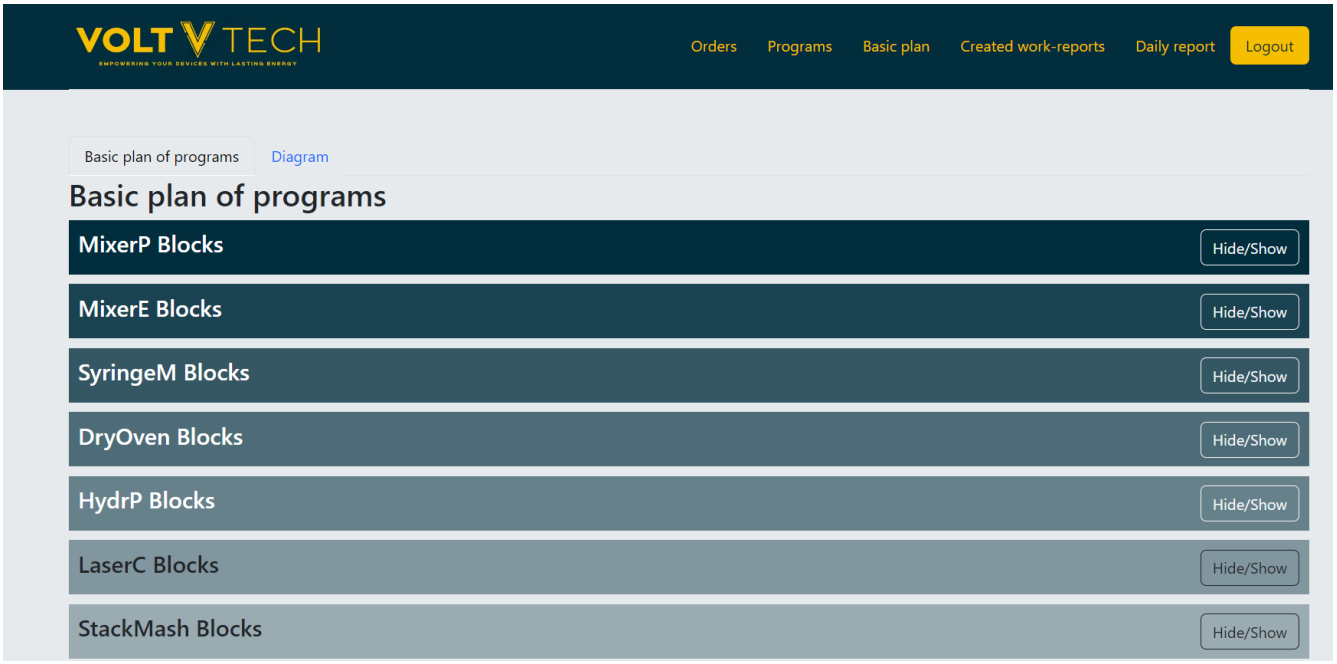


Рисунок А.7 – Вигляд вкладки Basic plan of programs

Вигляд вкладки Basic plan of programs при показі інформації одного з блоків і підсвічуванні його іншим кольором наведено на рисунку А.8.

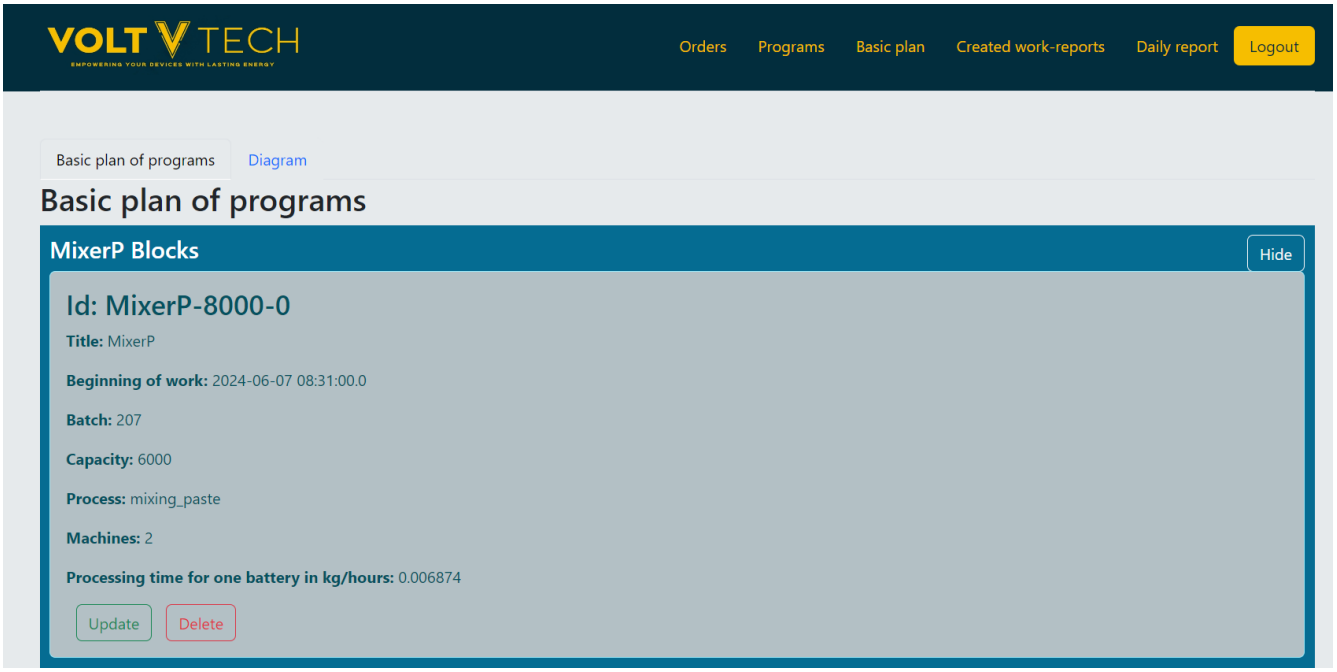


Рисунок А.8 – Вигляд вкладки з відкритим блоком MixerP

Вигляд сторінки для редагування програми, яка з’являється при натисканні на кнопку Update показано на рисунку А.9

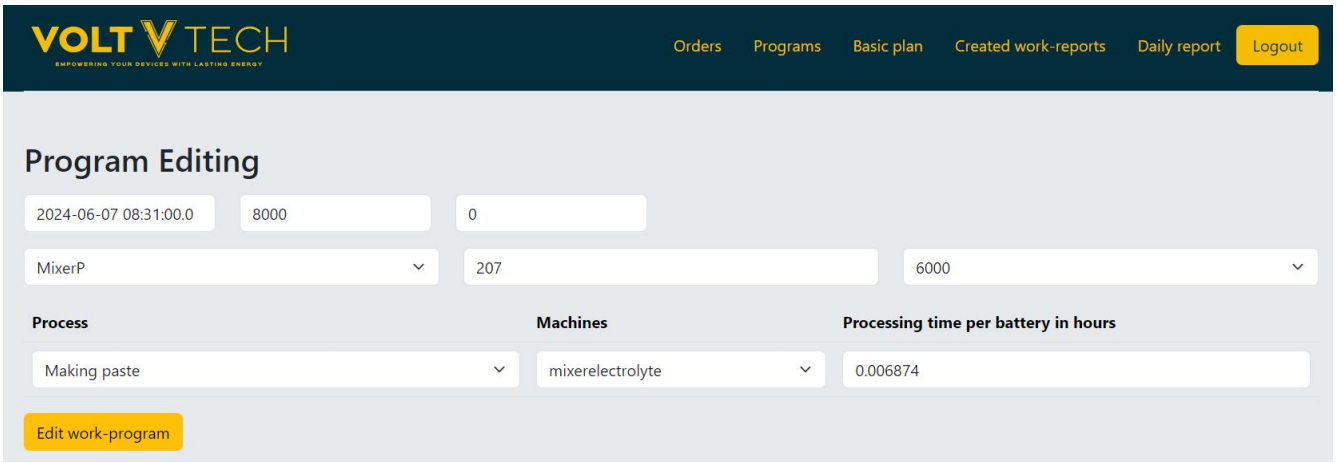


Рисунок А.9 – Вигляд сторінки редагування програми

Вигляд внутрішньої вкладки Diagram при виборі одного з блоків наведено на рисунку А.10.

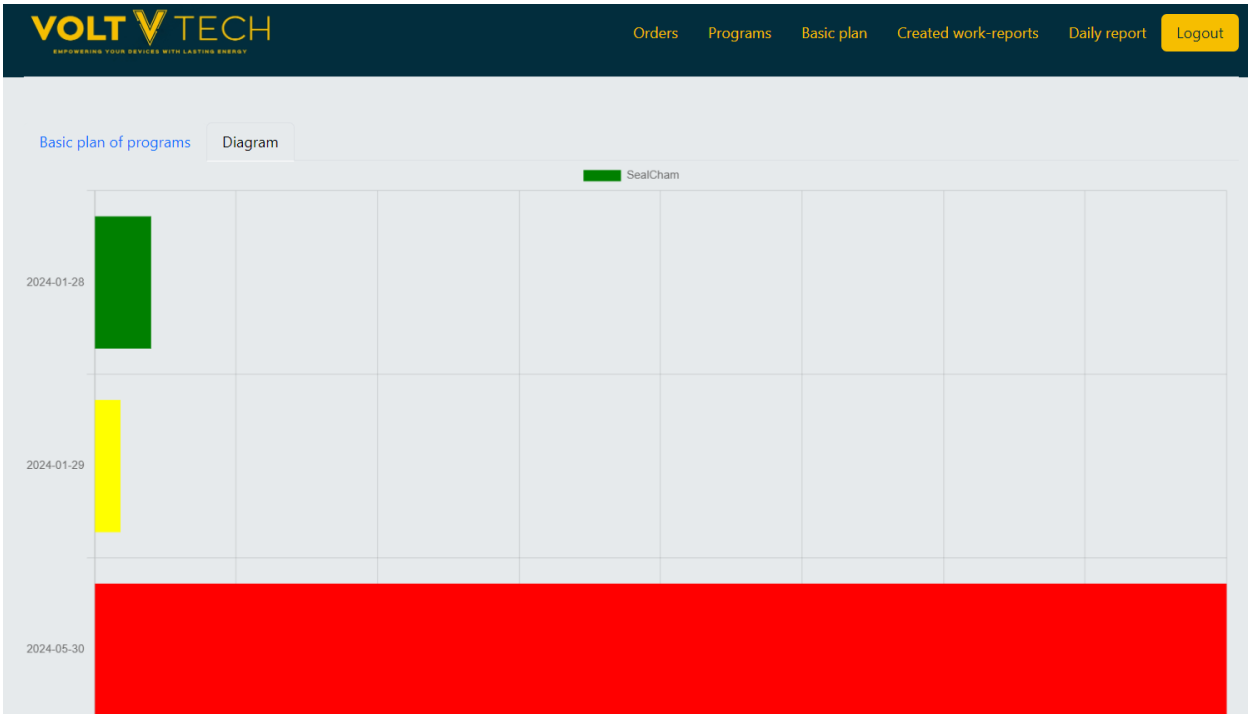
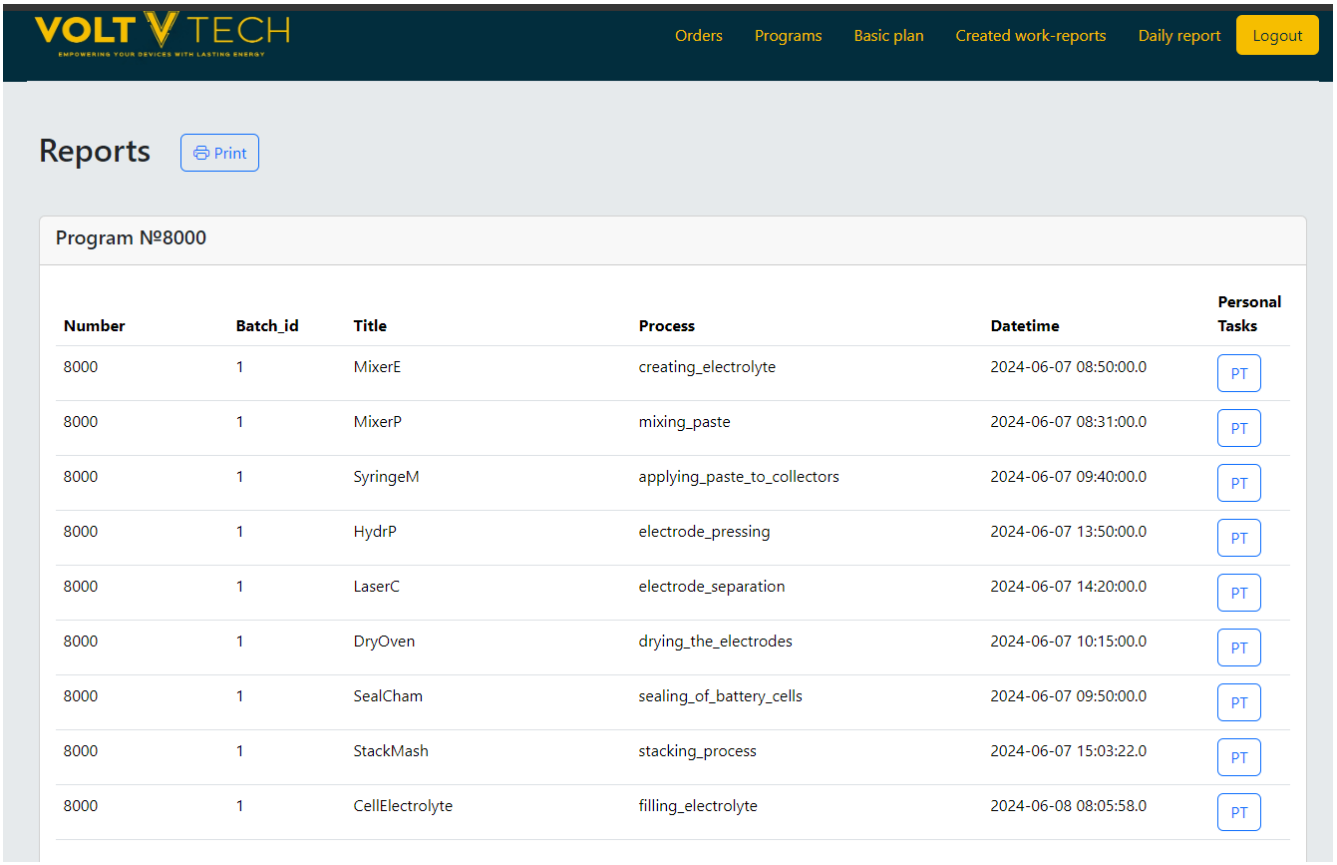


Рисунок А.10 – Вигляд графіку завантаженості для одного з блоків машин.

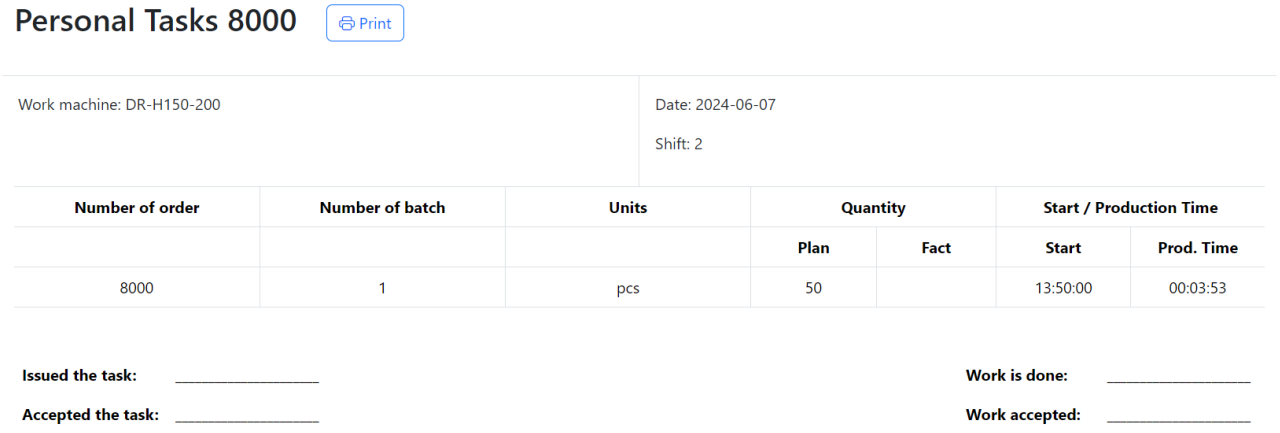
3.2.4 Перегляд створених програм для одного замовлення

На вкладці Created work-reports користувач може переглядати створені програми, які групуються в одне замовлення за номером. Також при натисканні на кнопку PT доступні персональні завдання для кожної машини, які розкривають деталі роботи. Їх можна роздрукувати при натисканні на кнопку Print. Вигляд вкладки Created work-reports представлено на рисунку А.11. Вигляд сторінки з персональними завданнями, яка з’являється при натисканні на кнопку PT відображено на рисунку А.12.



Number	Batch_id	Title	Process	Datetime	Personal Tasks
8000	1	MixerE	creating_electrolyte	2024-06-07 08:50:00.0	PT
8000	1	MixerP	mixing_paste	2024-06-07 08:31:00.0	PT
8000	1	SyringeM	applying_paste_to_collectors	2024-06-07 09:40:00.0	PT
8000	1	HydrP	electrode_pressing	2024-06-07 13:50:00.0	PT
8000	1	LaserC	electrode_separation	2024-06-07 14:20:00.0	PT
8000	1	DryOven	drying_the_electrodes	2024-06-07 10:15:00.0	PT
8000	1	SealCham	sealing_of_battery_cells	2024-06-07 09:50:00.0	PT
8000	1	StackMash	stacking_process	2024-06-07 15:03:22.0	PT
8000	1	CellElectrolyte	filling_electrolyte	2024-06-08 08:05:58.0	PT

Рисунок А.11 – Вигляд вкладки Created work-reports для одного замовлення



Number of order	Number of batch	Units	Quantity		Start / Production Time	
			Plan	Fact	Start	Prod. Time
8000	1	pcs	50		13:50:00	00:03:53

Issued the task: _____ **Work is done:** _____

Accepted the task: _____ **Work accepted:** _____

Рисунок А.12 – Вигляд сторінки персонального завдання для однієї з машин

3.2.5 Перегляд денного рапорту

На вкладці Daily report користувач може побачити виконані робочі програми за сьогодні, інформацію про проведений ремонт. Цю сторінку можна роздрукувати при натисканні на кнопку Print. Проєктувальник використовує інформацію про витрачений час на ремонт для проєктування наступних робочих програм. Вигляд вкладки Daily report відображено на рисунку А.13.

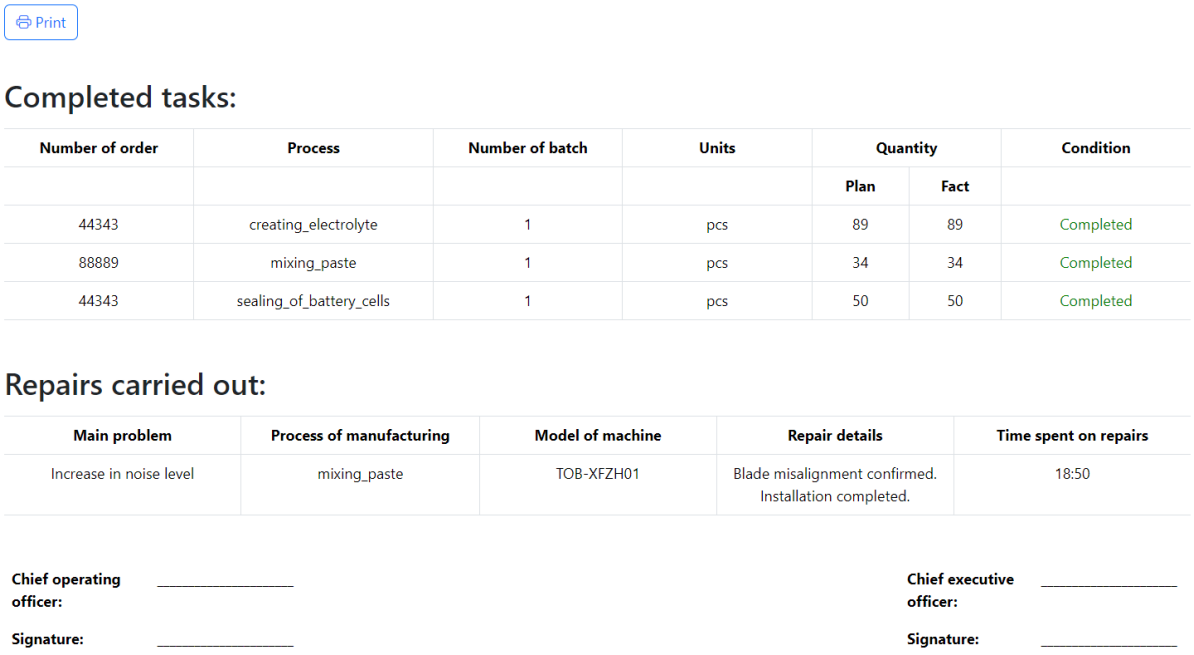


Рисунок А.13 – Вигляд сторінки Daily reports

3.3 Опис функціоналу для користувача Диспетчер

3.3.1 Перегляд програм, запланованих на поточний день

На головній сторінці додатку при вході з боку диспетчера одразу відкриваються програми, виконання яких заплановано проєктувальником на поточний день (рис. А.4). Також доступна кнопка PT, яка показує деталі роботи з програмою (рис.А.12).

3.3.2 Моніторинг показників підприємства

На вкладці Monitoring диспетчер може запустити виробничі процеси натисканням на кнопку Start (рис. А.14). Після цього розпочинається виконання запланованих на поточний день програм, про що говорять повідомлення-тости внизу екрану. Хід процесів можна відслідковувати як на таблиці з загальними показниками, так і на детальних таблицях до кожної машини.

При появі неполадок програма відображає їх червоним кольором у відповідних комірках таблиць та тостом внизу екрану (рис. А.15). Диспетчер повинен зупинити процеси натисканням на кнопку Pause, зафіксувати показники поломки та створити рапорт на ремонт.

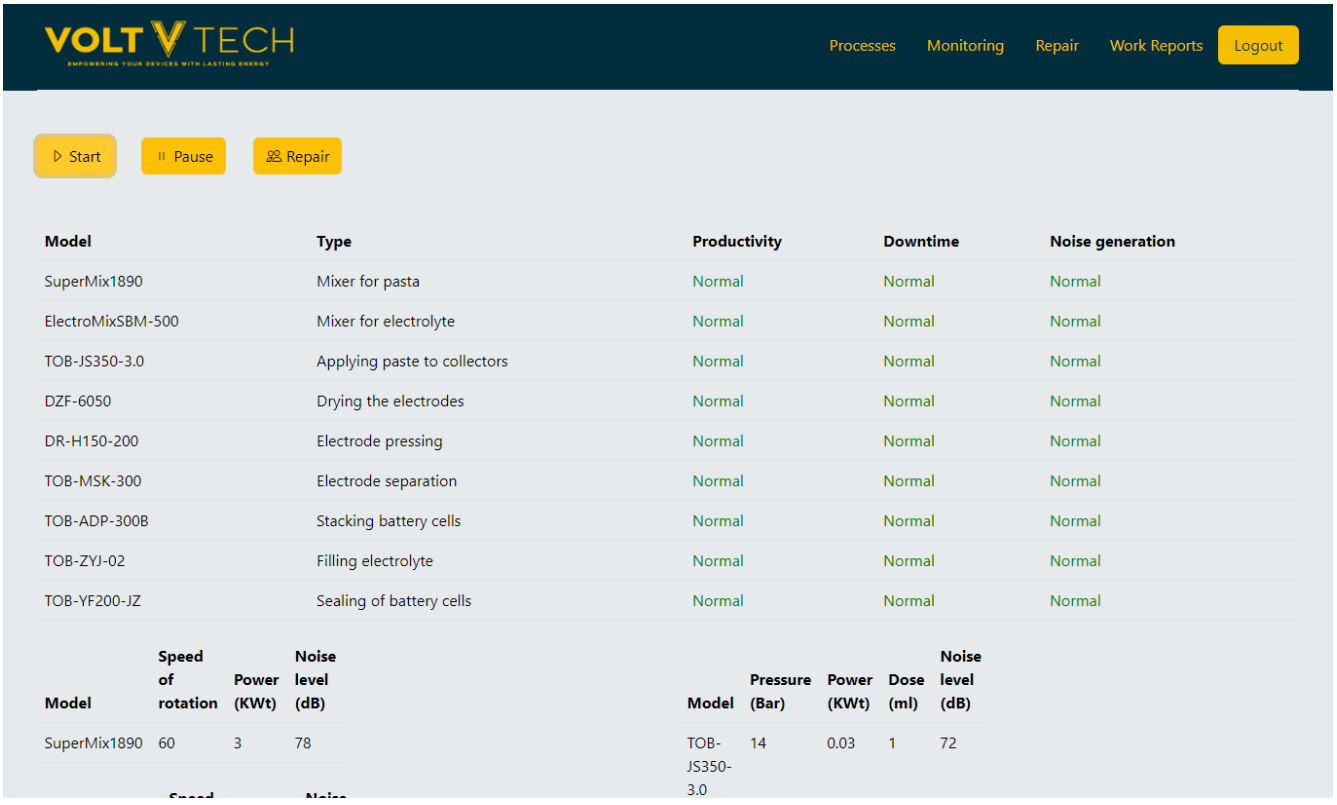


Рисунок А.14 – Вигляд вкладки Monitoring

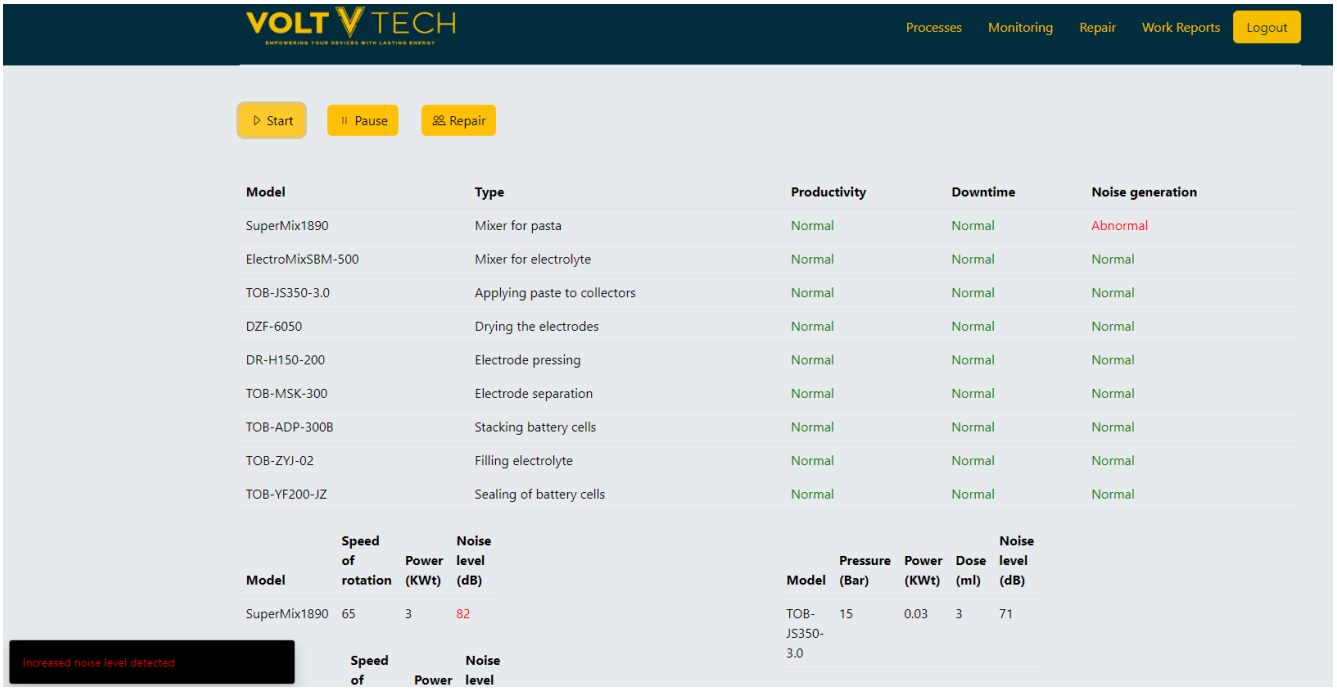


Рисунок А.15 – Вигляд вкладки при виникненні неполадки

Перехід на сторінку для створення рапорту здійнюється за допомогою натискання на кнопку Repair (рис. А.16). На сторінці є такі поля для заповнення:

– Enter main problem: поле для занесення головної причини створення рапорту;

– Select process: заповнення даних про процес, на якому відбулась поломка, можна обрати з випадаючого списку;

– Select a model: заповнення даних про модель машини, на якій відбулась поломка, можна обрати з випадаючого списку;

– Enter the lower limit of the range of acceptable values: поле для введення нижньої границі діапазону прийнятних значень для даного показника машини;

– Enter the upper limit of the range of acceptable values: поле для введення верхньої границі діапазону прийнятних значень для даного показника машини;

– Enter abnormal value: поле для введення не нормованого значення;

– Enter the possible cause of the program: Поле для введення можливої причини неправності.

Після створення рапорту натисканням кнопки Create report програма перенаправлює користувача на вкладку Repair, де відображається створений ним рапорт у статусі Відправлено (SENT).

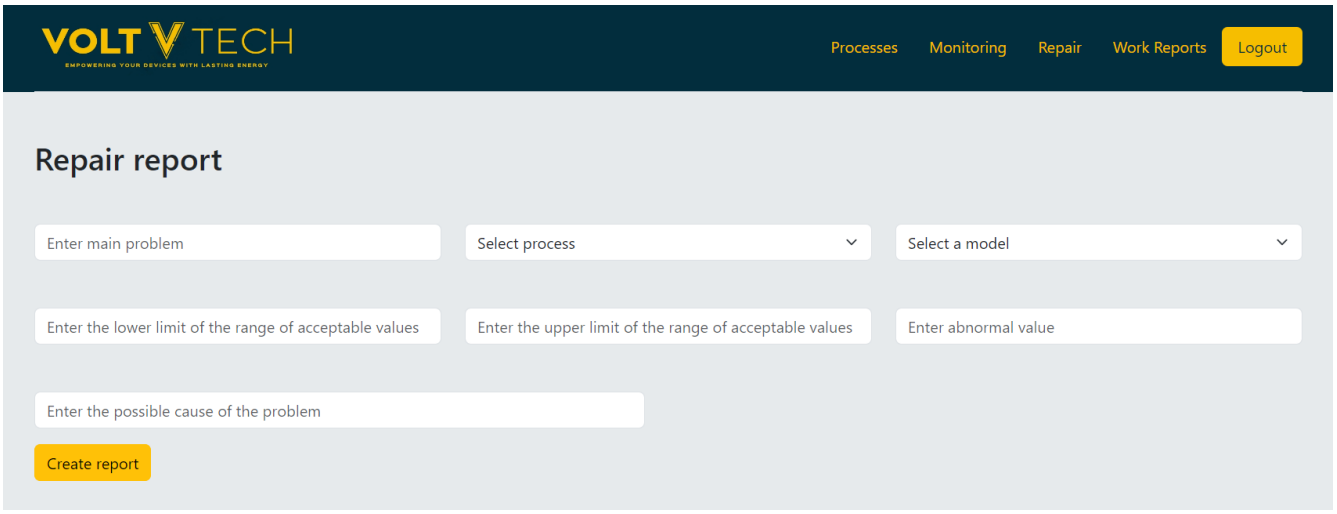


Рисунок А.16 – Вигляд сторінки для створення рапорту

3.3.3 Перегляд статусів рапорту

На вкладці Repair диспетчер може переглядати створені ним рапорти на ремонт і відстежувати, коли стптус Відправлено зміниться на Виконано (COMPLETED). Вигляд вкладки Repair відображено на рисунку А.17.

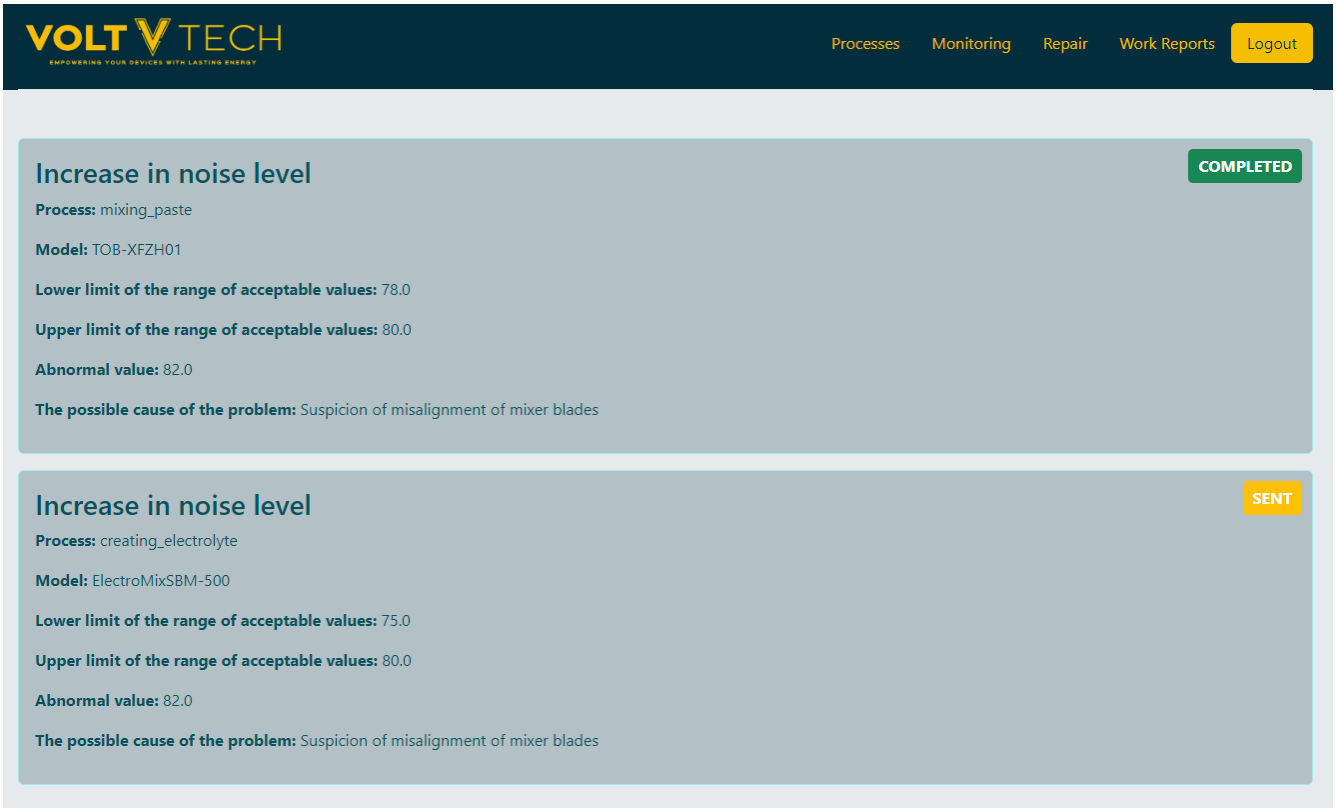


Рисунок А.17 – Вигляд вкладки Repair

3.3.4 Створення денного рапорту

На вкладці Daily report користувач може побачити виконані робочі програми за сьогодні, інформацію про проведений ремонт. Цю сторінку можна роздрукувати при натисканні на кнопку Print (рис. А.13).

3.4 Опис функціоналу для користувача Бригадир ремонтної бригади

3.4.1 Редагування рапорту

На головній сторінці додатку при вході з боку бригадира відкриваються створені диспетчером рапорти на ремонт (рис. А.5). На рапортах зі статусом SENT бригадиру доступна кнопка Supplement the report при натисканні на яку відкривається сторінка редагування рапортів (рис. А.18). Бригадир повинен

заповнювати вказані на ній поля лише після проведення ремонту. Поля на сторінці редагування такі:

– Enter repair details: поле для внесення деталей щодо ремонту, підтвердження або спростовування гіпотези диспетчера щодо поломки;

– Time spent on repairs: поле для внесення часу, витраченого на ремонт.

Після занесених змін бригадир повинен натиснути на кнопку Update report для збереження. Рапорт відображається на вкладці зі статусом Виконано (COMPLETED). Зміна статусу також відображається і у диспетчера.

Рисунок А.18 – Сторінка для редагування рапорту

3.4.2 Зміна статусів машин

На вкладці Replacing machines диспетчер може переглядати усі наявні машини на підприємстві та змінювати їх статуси в залежності від того, чи активна ця машина зараз (ри. А.19). Статус USED означає, що машина працює на підприємстві, STORAGE – що вона знаходиться на складі. Змінювати статуси машин можна лише після проведення ремонту з заміною машини іншою. Зміна можлива при натисканні на випадаючий список, виборі потрібного статусу та збереження змін за допомогою кнопки Save changes. При успішній заміні машини зв’явиться повідомлення-тост про це внизу екрану.

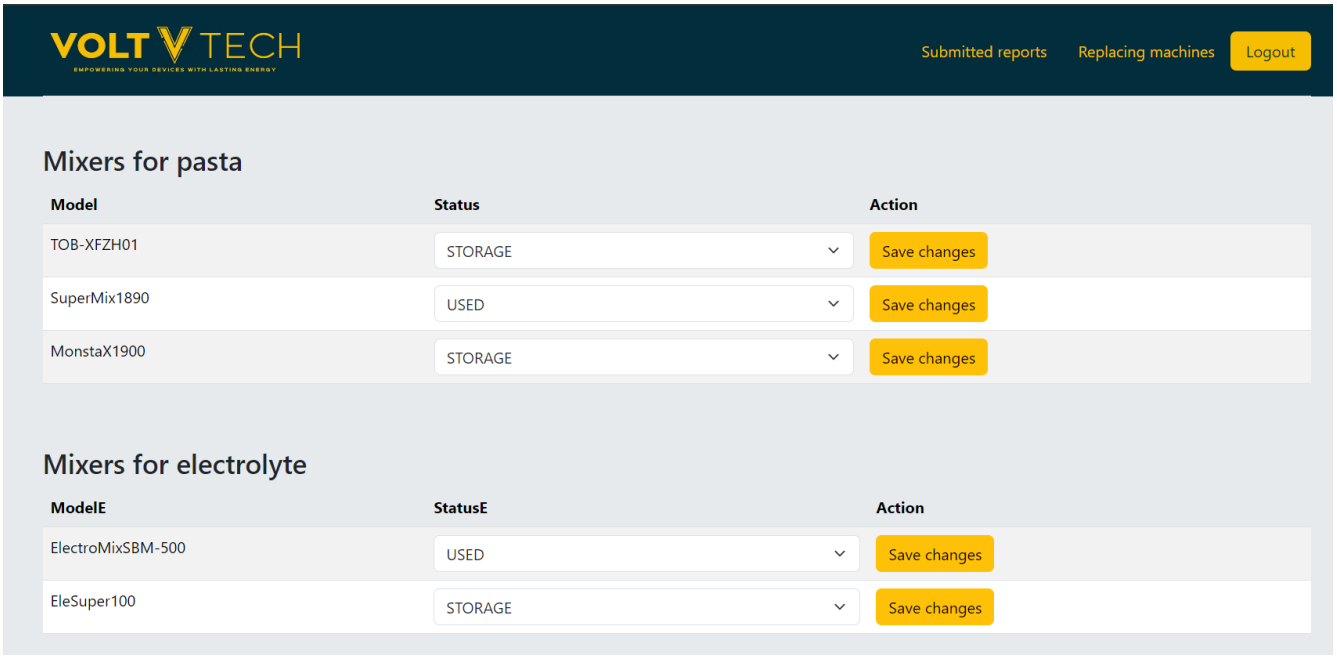


Рисунок А.19 – Вигляд вкладки Replacing machines

4 АВАРІЙНІ СИТУАЦІЇ

При виникненні збоїв в завантаженні сторінок програми працівник повинен перевірити:

– наявність підключення локального серверу;

– наявність підключення до мережі Інтернет;

– наявність запущеної програми у середовищі розробки;

При виникненні збоїв при введенні даних необхідно перевірити, чи всі поля для введення коректно заповнені текстом;

При виникненні збоїв при завантаженні графіків імітації процесу у проєктувальника необхідно перевірити, чи був обраний один блок для відображення графіку;

При виникненні збоїв при завантаженні документу на друк (наприклад, на друці відсутні налаштування для прийнятного візуального відображення):

– вийти з меню відправки документа на друк;

– перезавантажити сторінку;

– знову натиснути на кнопку Print.

При некоректній поведінці сторінок програми необхідно перезавантажити її з очищенням кешу браузера.

ДОДАТОК Б

Текст програми

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-parent</artifactId>
       <version>3.2.2</version>
       <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>Manufac</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Manufac</name>
    <description>Manufacturing of something</description>
    <properties>
       <java.version>17</java.version>
    </properties>

    <dependencies>
       <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-thymeleaf</artifactId>
       </dependency>
       <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-web</artifactId>
       </dependency>
       <dependency>
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
          <version>8.0.33</version>
          <scope>runtime</scope>
       </dependency>
       <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-data-jpa</artifactId>
       </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

ProgramController:

```
package com.example.Manufac.controllers;
import com.example.Manufac.models.Machines;
import com.example.Manufac.models.WorkProgram;
import com.example.Manufac.repo.MachinesRepository;
import com.example.Manufac.repo.WorkProgramRepository;
import jakarta.annotation.security.PermitAll;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.math.BigInteger;
import java.sql.Timestamp;
import java.util.List;
import java.util.Optional;
@Controller
public class ProgramController {
    @Autowired
    private WorkProgramRepository WorkProgramRepository;

    @Autowired
    private MachinesRepository MachinesRepository;

    public ProgramController(MachinesRepository machinesRepository) {
        this.MachinesRepository = machinesRepository;}
    @GetMapping("/program")
    public String WorkProg(Model model) {
        List<Machines> activeModels =
MachinesRepository.findActiveModels();
        model.addAttribute("machines", activeModels);
        return "work-prog";
    }

    @PostMapping("/program")
    public String workProgramPost(@RequestParam int number, @RequestParam
int morenumber, @RequestParam String title, @RequestParam int batch,
```

```
 @RequestParam String process,
                                    @RequestParam BigInteger machines,
@RequestParam String capacity, @RequestParam float onebatt_hour,
@RequestParam String datetime, Model model) {
        Timestamp timestamp = Timestamp.valueOf(datetime);
        WorkProgram WorkProgram = new
WorkProgram(number,morenumber,title,batch,process,machines,capacity,onebatt
_hour,timestamp);
        WorkProgramRepository.save(WorkProgram);
        return "redirect:/base-plan";
    }

    @GetMapping("/program/{id}/edit")
    public String editProgram(@PathVariable("id") Long id, Model model) {
        Optional<WorkProgram> optionalWorkProgram =
WorkProgramRepository.findById(id);
        if (optionalWorkProgram.isPresent()) {
            WorkProgram workProgram = optionalWorkProgram.get();
            List<Machines> activeModels =
MachinesRepository.findActiveModels();
            model.addAttribute("workProgram", workProgram);
            model.addAttribute("machines", activeModels);
            return "program-edit";         } else {

            return "redirect:/base-plan";
        }
    }

    @PostMapping("/program/{id}/edit")
    public String workProgramPostEdit(@PathVariable("id") Long id,
@RequestParam int number, @RequestParam int morenumber, @RequestParam
String title, @RequestParam int batch, @RequestParam String process,
                                    @RequestParam BigInteger machines,
@RequestParam String capacity, @RequestParam float onebatt_hour,
@RequestParam String datetime, Model model) {
        WorkProgram WorkProgram =
WorkProgramRepository.findById(id).orElseThrow();
        Timestamp timestamp = Timestamp.valueOf(datetime);
        WorkProgram.setNumber(number);
        WorkProgram.setMorenumber(morenumber);
        WorkProgram.setTitle(title);
        WorkProgram.setBatch(batch);
```

```
         WorkProgram.setProcess(process);
        WorkProgram.setMachines(machines);
        WorkProgram.setCapacity(capacity);
        WorkProgram.setOnebatt_hour(onebatt_hour);
        WorkProgram.setDatetime(timestamp);
        WorkProgramRepository.save(WorkProgram);
        return "redirect:/base-plan";
    }

    @PostMapping("/program/{id}/remove")
    public String workProgramPostDelete(@PathVariable("id") Long id, Model
model) {
        WorkProgram WorkProgram =
WorkProgramRepository.findById(id).orElseThrow();
        WorkProgramRepository.delete(WorkProgram);
        return "redirect:/base-plan";
    }
}
```

MonitoringController:

```
package com.example.Manufac.controllers;
import com.example.Manufac.models.*;
import com.example.Manufac.repo.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import java.sql.Timestamp;
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;
@Controller
public class MonitoringController {
    @Autowired
    private MixerpastaRepository mixerpastaRepository;
    @Autowired
    private MixerelectrolyteRepository mixerelectrolyteRepository;
    @Autowired
```

```
    private SyringemachinesRepository syringemachinesRepository;
    @Autowired
    private DryovenRepository dryovenRepository;
    @Autowired
    private HydrpressRepository hydrpressRepository;
    @Autowired
    private LasercutterRepository lasercutterRepository;
    @Autowired
    private StackmashRepository stackmashRepository;
    @Autowired
    private CellelectrolytemRepository cellelectrolytemRepository;
    @Autowired
    private SealingchamberRepository sealingchamberRepository;
    @Autowired
    private com.example.Manufac.repo.WorkProgramRepository
workProgramRepository;
    @Autowired
    private MachinesRepository MachinesRepository;
    public MonitoringController(MachinesRepository machinesRepository) {
        this.MachinesRepository = machinesRepository;}
    @GetMapping("/monitoring")
    public String showMonitoring (Model model) {
        List<Object[]> findMachinesP =
mixerpastaRepository.findUsedMachines();
        model.addAttribute("mixers", findMachinesP);
        List<Object[]> findMachinesE =
mixerelectrolyteRepository.findUsedMachinesE();
        model.addAttribute("mixere", findMachinesE);
        List<Object[]> findMachinesS =
syringemachinesRepository.findUsedMachinesS();
        model.addAttribute("syrm", findMachinesS);
        List<Object[]> findMachinesD =
dryovenRepository.findUsedMachinesD();
        model.addAttribute("dryo", findMachinesD);
        List<Object[]> findMachinesH =
hydrpressRepository.findUsedMachinesH();
        model.addAttribute("hydr", findMachinesH);
        List<Object[]> findMachinesL =
lasercutterRepository.findUsedMachinesL();
        model.addAttribute("laser", findMachinesL);
        List<Object[]> findMachinesST =
stackmashRepository.findUsedMachinesST();
```

```
            model.addAttribute("stack", findMachinesST);
        List<Object[]> findMachinesC =
cellelectrolytemRepository.findUsedMachinesC();
        model.addAttribute("elec", findMachinesC);
        List<Object[]> findMachinesSC =
sealingchamberRepository.findUsedMachinesSC();
        model.addAttribute("seal", findMachinesSC);
        LocalDate currentDate = LocalDate.now();
        Timestamp startOfDay =
Timestamp.valueOf(currentDate.atStartOfDay());
        Timestamp endOfDay = Timestamp.valueOf(currentDate.atTime(23, 59,
59));
        List<WorkProgram> wprogram =
workProgramRepository.findByDatetimeBetween(startOfDay, endOfDay);
        List<String> processNames = wprogram.stream()
                .map(WorkProgram::getProcess)
                .collect(Collectors.toList());
        model.addAttribute("wprogram", wprogram);
        model.addAttribute("processNames", processNames);
        return "monitoring";     }
    @GetMapping("/monitoring/repair")
    public String showRepairPage(Model model) {
        List<Machines> activeModels =
MachinesRepository.findActiveModels();
        model.addAttribute("machines", activeModels);
        return "repair";
    }

}
```

ReplaceMachinesController:

```
package com.example.Manufac.controllers;
import com.example.Manufac.models.*;
import com.example.Manufac.repo.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@Controller
public class ReplaceMachinesController {
```

```
    @Autowired
    private MixerpastaRepository mixerpastaRepository;
    @Autowired
    private MixerelectrolyteRepository mixerelectrolyteRepository;
    @Autowired
    private SyringemachinesRepository syringemachinesRepository;
    @Autowired
    private DryovenRepository dryovenRepository;
    @Autowired
    private HydrpressRepository hydrpressRepository;
    @Autowired
    private LasercutterRepository lasercutterRepository;
    @Autowired
    private StackmashRepository stackmashRepository;
    @Autowired
    private CellelectrolytemRepository cellelectrolytemRepository;
    @Autowired
    private SealingchamberRepository sealingchamberRepository;
    @GetMapping("/replace-mach")
    public String showReplaceMach(Model model) {
        List<Object[]> mixerpastaList =
mixerpastaRepository.findMachines();
        model.addAttribute("mixerpastaList", mixerpastaList);
        List<Object[]> mixerelectrolyteList =
mixerelectrolyteRepository.findMachinesE();
        model.addAttribute("mixerelectrolyteList", mixerelectrolyteList);
        List<Object[]> syringemachinesList =
syringemachinesRepository.findMachinesS();
        model.addAttribute("syringemachinesList", syringemachinesList);
        List<Object[]> dryovenList = dryovenRepository.findMachinesD();
        model.addAttribute("dryovenList", dryovenList);
        List<Object[]> hydrpressList = hydrpressRepository.findMachinesH();
        model.addAttribute("hydrpressList", hydrpressList);
        List<Object[]> lasercutterList =
lasercutterRepository.findMachinesL();
        model.addAttribute("lasercutterList", lasercutterList);
        List<Object[]> stackmashList =
stackmashRepository.findMachinesST();
        model.addAttribute("stackmashList", stackmashList);
        List<Object[]> cellelectrolytemList =
cellelectrolytemRepository.findMachinesC();
```

```
            model.addAttribute("cellelectrolytemList",
cellelectrolytemList);
        List<Object[]> sealingchamberList =
sealingchamberRepository.findMachinesSC();
        model.addAttribute("sealingchamberList", sealingchamberList);
        return "replace-mach";
    }
    @PostMapping("/update-status")
    @ResponseBody
    public String updateStatus(@RequestParam("id") BigInteger id,
@RequestParam("status") String status) {
                Mixerpasta mixerpasta =
mixerpastaRepository.findById(id).orElseThrow(() -> new
RuntimeException("Mixerpasta not found"));
                mixerpasta.setStatus(Mixerpasta.Status.valueOf(status));
mixerpastaRepository.save(mixerpasta);
                return "Status updated successfully";
    }
    @PostMapping("/update-statusE")
    @ResponseBody
    public String updateStatusE(@RequestParam("id") BigInteger id,
@RequestParam("statusE") String statusE) {
        Mixerelectrolyte mixerelectrolyte =
mixerelectrolyteRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Mixerelectrolyte
not found"));
    mixerelectrolyte.setStatus(Mixerelectrolyte.StatusE.valueOf(statusE));
        mixerelectrolyteRepository.save(mixerelectrolyte);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusS")
    @ResponseBody
    public String updateStatusS(@RequestParam("id") BigInteger id,
@RequestParam("statusS") String statusS) {
        Syringemachines syringemachines =
syringemachinesRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Syringemachines
not found"));
      syringemachines.setStatusS(Syringemachines.StatusS.valueOf(statusS));
        syringemachinesRepository.save(syringemachines);
        return "Status updated successfully";
```

```
}

    @PostMapping("/update-statusD")
    @ResponseBody
    public String updateStatusD(@RequestParam("id") BigInteger id,
@RequestParam("statusD") String statusD) {
        Dryoven dryoven = dryovenRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Dryoven not
found"));
        dryoven.setStatusD(Dryoven.StatusD.valueOf(statusD));
        dryovenRepository.save(dryoven);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusH")
    @ResponseBody
    public String updateStatusH(@RequestParam("id") BigInteger id,
@RequestParam("statusH") String statusH) {
        Hydrpress hydrpress = hydrpressRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Hydrpress not
found"));
        hydrpress.setStatusH(Hydrpress.StatusH.valueOf(statusH));
        hydrpressRepository.save(hydrpress);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusL")
    @ResponseBody
    public String updateStatusL(@RequestParam("id") BigInteger id,
@RequestParam("statusL") String statusL) {
        Lasercutter lasercutter = lasercutterRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Lasercutter not
found"));
        lasercutter.setStatusL(Lasercutter.StatusL.valueOf(statusL));
        lasercutterRepository.save(lasercutter);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusST")
    @ResponseBody
    public String updateStatusST(@RequestParam("id") BigInteger id,
@RequestParam("statusST") String statusST) {
        Stackmash stackmash = stackmashRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Stackmash not
found"));
```

```
        stackmash.setStatusST(Stackmash.StatusST.valueOf(statusST));
        stackmashRepository.save(stackmash);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusC")
    @ResponseBody
    public String updateStatusC(@RequestParam("id") BigInteger id,
@RequestParam("statusC") String statusC) {
        Cellelectrolytem cellelectrolytem =
cellelectrolytemRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Cellelectrolytem
not found"));

cellelectrolytem.setStatusC(Cellelectrolytem.StatusC.valueOf(statusC));
        cellelectrolytemRepository.save(cellelectrolytem);
        return "Status updated successfully";
    }
    @PostMapping("/update-statusSC")
    @ResponseBody
    public String updateStatusSC(@RequestParam("id") BigInteger id,
@RequestParam("statusSC") String statusSC) {
        Sealingchamber sealingchamber =
sealingchamberRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Sealingchamber not
found"));

sealingchamber.setStatusSC(Sealingchamber.StatusSC.valueOf(statusSC));
        sealingchamberRepository.save(sealingchamber);
        return "Status updated successfully";
    }
}
```

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК 506100.018 ПЗ	Пояснювальна записка	114 с., ф. А4
2		Керівництво користувача	23 с., вкл. до ПЗ
3		Текст програми	12 с., вкл. до ПЗ
		Інші документи	
4		Демонстраційний матеріал	у форматі
			*.pptx

Змін	Арк.	№ документа	Підпис	Дата	ГЮИК 506100.018 ВД		
Розроб.		Вінниченко			Розробка компоненту MES-сисиеми для оптимізації виробничих процесів підприємства Відомість кваліфікаційної роботи	Літера	Аркуш	Аркушів
Перев.		Колесник				Н	1	1
						ХНУРЕ Кафедра КІТАР		
Н.конт.		Бронніков						
Затвер.		Невлюдов						