ДОДАТОК А

Код програми

Клієнтський код форми додавання зони мовою TypeScript:

```
'use client';

import { useState } from 'react';
import { Box, Button, Alert, CircularProgress } from '@mui/material';
import {
  SubmitHandler,
  useForm,
  Controller,
  useFormState,
} from 'react-hook-form';

import GlobalInput from '@/components/ElementsAndBlocks/GlobalInput';

import { useShelvesStore } from '@/store/shelvesStore';

import { addShelve } from '@/services/shelves/shelvesService';
import { addReport } from '@/services/reports/reportsService';

import { getQueryByNameFromUrl } from '@/helpers/locationHelpers';

import { qSKeys } from '@/constants/qSKeys';

import { IAddShelveReqBody } from '@/services/shelves/shelvesTypes';
import { TError } from '@/types/error';
import { generateReportBody } from '@/helpers/reportHelpers';

interface IAddShelveFormProps {
  popupState: any;
}
```

```
const AddShelveForm: React.FC<IAddShelveFormProps> = ({ popupState }) => {
  const { getAllShelvesAction, getShelveByIDAction, allShelves } =
    useShelvesStore();

  const { handleSubmit, control } = useForm<IAddShelveReqBody>();
  const { errors } = useFormState({
    control,
  });

  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState('');

  const onSubmit: SubmitHandler<IAddShelveReqBody> = async ({
    shelveID,
    width,
    height,
    length,
    maxShelvesCount,
  }) => {
    try {
      setIsLoading(true);
      setError('');

      const { data: shelve } = await addShelve({
        shelveID,
        width,
        height,
        length,
        maxShelvesCount,
      });
      const shelveSearchVal = getQueryByNameFromUrl(qSKeys.shelveSearch);

      if (shelve && shelveSearchVal)
        getShelveByIDAction(Number(shelveSearchVal));
```

```
    else if (shelve && !shelveSearchVal) getAllShelvesAction();

    popupState.close();

    const addShelveReport = generateReportBody(
      `Add shelve with id = ${shelveID}`,
    );
    await addReport(addShelveReport);
  } catch (err: TError) {
    const error = err?.message ? err.message : 'Get shelve error';

    setError(error);
  } finally {
    setIsLoading(isLoading);
  }
};

const maxShelveID = allShelves.reduce((maxID, shelve) => {
  return Math.max(maxID, shelve.shelveID);
}, 0);

return (
  <form onSubmit={handleSubmit(onSubmit)}>
    <Controller
      control={control}
      name='shelveID'
      rules={numberRules}
      defaultValue={maxShelveID + 1}
      render={({ field: { onChange, value } }: any) => (
        <GlobalInput
          value={value}
          onChange={onChange}
          required
          label='Shelve id'
          type='number'
```

```
      placeholder='0000'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.shelveID?.message}
    />
  )}
/>
<Controller
  control={control}
  name='width'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='Shelve Width'
      type='number'
      placeholder='00'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.width?.message}
    />
  )}
/>
<Controller
  control={control}
  name='height'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='Shelve Height'
      type='number'
      placeholder='00'
```

```
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.height?.message}
    />
  )}
/>
<Controller
  control={control}
  name='length'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='Shelve Length'
      type='number'
      placeholder='00'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.length?.message}
    />
  )}
/>
<Controller
  control={control}
  name='maxShelvesCount'
  defaultValue={allShelves?.length + 1}
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='New shelves count'
      type='number'
      placeholder='0000'
```

```
          customStyle={{ marginBottom: 3 }}
          helperText={errors?.maxShelvesCount?.message}
        />
      )}
    />

    {!isLoading && error && (
      <Alert severity='error' sx={{ marginBottom: 2 }}>
        {error}
      </Alert>
    )}

    <Box sx={{ width: '100%', display: 'flex', justifyContent: 'flex-end' }}>
      <Button
        variant='contained'
        size='medium'
        type='submit'
        sx={{ minWidth: '100px' }}
      >
        Add
        {isLoading && (
          <CircularProgress
            color='secondary'
            sx={{ marginLeft: 1 }}
            size={25}
          />
        )}
      </Button>
    </Box>
  </form>
 );
};

const numberRules = {
 required: {
```

```
    value: true,
    message: 'This field is required',
  },
  pattern: {
    value: /^\d+$/,
    message: 'Data is not correct',
  },
};

export default AddShelveForm;
```

Серверний код додавання зони мовою TypeScript:

```
async addShelve(req: Request, res: Response) {
  try {
    const errors = validationResult(req);

    if (!errors.isEmpty())
      res.status(400).json({
        message: 'Data is not correct',
        errors: errors.array(),
      });

    const { shelveID, width, height, length, maxShelvesCount } = req.body;

    const savedShelve: IShelve | null | any = await shelveRepository.findOne<{
      shelveID: number;
    }>({
      shelveID,
    });

    if (savedShelve?.shelveID)
      return res
        .status(404)
        .json({ message: `Shelve with id ${shelveID} is already exist` });

    const { newX, newY } =
```

```
      await shelveService.getNewCoordinates(maxShelvesCount);

    const newShelve = await shelveRepository.create({
      shelveID,
      shelveDimensions: { width, height, length },
      coordinates: {
        x: newX,
        y: newY,
      },
      products: [],
      percentBusyVolume: 0,
    });

    res.send(newShelve);
  } catch (err) {
    return res.status(500).json(err);
  }
}
```

Клієнтський код картки зони з функціоналом її видалення мовою TypeScript:

```
'use client';

import {
  Alert,
  Box,
  Button,
  Card,
  CardContent,
  CircularProgress,
  Typography,
} from '@mui/material';

import { useAuthStore } from '@/store/authStore';

import { basicTheme } from '@/theme/theme';
```

```
import { ICoordinates, IDimensions } from '@/types/sizeAndCoordinates';
import React, { useState } from 'react';
import RemoveShelveBtn from '@/pages/ShelvesPage/components/RemoveShelveBtn';

interface IShelveCardProps {
  shelveID: number;
  shelveDimensions: IDimensions;
  coordinates: ICoordinates;
  percentBusyVolume: number;
}

const ShelveCard: React.FC<IShelveCardProps> = ({
  shelveID,
  shelveDimensions,
  coordinates,
  percentBusyVolume,
}) => {
  const { user } = useAuthStore();

  const [isLoading, setIsLoading] = useState(false);

  const [error, setError] = useState('');

  const percentEmptyVolume = 100 - percentBusyVolume;

  return (
    <Card key={shelveID}>
      <CardContent
        sx={{
          background: `linear-gradient(transparent ${percentEmptyVolume}%,
${basicTheme.success} ${percentEmptyVolume}%)`,
        }}
      >
        <Typography component='h3' variant='h3'>
          Shelve id: {shelveID}
```

```
    </Typography>
    <Box sx={{ marginTop: 3 }}>
      <Typography component='h4' variant='h4'>
        Coordinates:
      </Typography>
      <Typography component='p' variant='body1'>
        x: {coordinates?.x}; y: {coordinates?.y};
      </Typography>
    </Box>
    <Box sx={{ marginTop: 3 }}>
      <Typography component='h4' variant='h4'>
        Shelve dimensions:
      </Typography>
      <Typography component='p' variant='body1'>
        width: {shelveDimensions?.width} sm;
        <br />
        length: {shelveDimensions?.length} sm;
        <br />
        height: {shelveDimensions?.height} sm;
      </Typography>
    </Box>
    {Boolean(user && user?.role === 'admin') && (
      <Box
        sx={{ marginTop: 3, display: 'flex', justifyContent: 'flex-end' }}
      >
        <RemoveShelveBtn shelveID={shelveID} setError={setError} />
      </Box>
    )}
    {error && (
      <Alert sx={{ marginTop: 2 }} severity='error'>
        {error}
      </Alert>
    )}
  </CardContent>
</Card>
```

```
  );
};

export default ShelveCard;



'use client';

import { useState } from 'react';
import {
  Popper,
  Button,
  Fade,
  Paper,
  Typography,
  Box,
  CircularProgress,
} from '@mui/material';
import PopupState, { bindToggle, bindPopper } from 'material-ui-popup-state';

import { useShelvesStore } from '@/store/shelvesStore';

import { removeProductByID } from '@/services/products/productsService';
import { addReport } from '@/services/reports/reportsService';

import { getQueryByNameFromUrl } from '@/helpers/locationHelpers';
import { generateReportBody } from '@/helpers/reportHelpers';

import { qSKeys } from '@/constants/qSKeys';
import { basicTheme } from '@/theme/theme';

import { TError } from '@/types/error';
import { removeShelveByID } from '@/services/shelves/shelvesService';

interface IRemoveShelveBtnProps {
```

```
  shelveID: number;
  setError: (error: string) => void;
}

const RemoveShelveBtn: React.FC<IRemoveShelveBtnProps> = ({
  shelveID,
  setError,
}) => {
  const { getShelveByIDAction, getAllShelvesAction } = useShelvesStore();

  const [isLoading, setIsLoading] = useState(false);

  const removeShelveHandler = async (popupState: any) => {
    try {
      setIsLoading(true);
      setError('');

      const { data: shelve } = await removeShelveByID(shelveID);

      const shelveSearchVal = getQueryByNameFromUrl(qSKeys.shelveSearch);

      if (shelve && shelveSearchVal)
        getShelveByIDAction(Number(shelveSearchVal));
      else if (shelve && !shelveSearchVal) getAllShelvesAction();

      const removeShelveReport = generateReportBody(
        `Remove shelve with id = ${shelveID}`,
      );
      await addReport(removeShelveReport);
    } catch (err: TError) {
      const error = err?.message ? err.message : 'Remove shelve error';

      setError(error);
    } finally {
      popupState.close();
```

```
    setIsLoading(false);
  }
};

return (
  <PopupState variant='popper' popupId='demo-popup-popper' disableAutoFocus>
    {(popupState) => (
      <div>
        <Button variant='contained' {...bindToggle(popupState)}>
          Remove
          {isLoading && (
            <CircularProgress
              color='secondary'
              sx={{ marginLeft: 1 }}
              size={25}
            />
          )}
        </Button>
        <Popper {...bindPopper(popupState)} transition placement='bottom-end'>
          {({ TransitionProps }) => (
            <Fade {...TransitionProps} timeout={350}>
              <Paper
                sx={{ marginTop: 1, marginBottom: 1, p: 2, minWidth: 450 }}
              >
                <Typography
                  component='h4'
                  variant='h4'
                  color={basicTheme.warning}
                >
                  Are you want to remove this shelve?
                </Typography>
                <Box sx={{ marginTop: 2, display: 'flex', gap: 2 }}>
                  <Button
                    sx={{ color: basicTheme.success }}
                    onClick={() => popupState.close()}
```

```
                >
                  No
                </Button>
                <Button
                  sx={{ color: basicTheme.error }}
                  onClick={() => removeShelveHandler(popupState)}
                >
                  Yes
                </Button>
              </Box>
            </Paper>
          </Fade>
        )}
      </Popper>
    </div>
  )}
 </PopupState>
 );
};

export default RemoveShelveBtn;
```

Серверний код видалення зони мовою TypeScript:

```
async deleteShelve(req: Request, res: Response) {
 try {
  const { id } = req.params;

  const shelve = await shelveRepository.getByID(Number(id));

  if (!shelve)
   return res.status(404).json({
    message: `Shelve with id = ${id} is not exist`,
    isShelveNotExist: true,
   });

  const shelveDeleteResult = await shelveRepository.deleteByID(Number(id));
```

```
    if (shelve?.products)
      await productRepository.deleteManyByIDs(shelve.products);

    return res.send(shelveDeleteResult);
  } catch (err) {
    return res.status(500).json(err);
  }
}
```

Клієнтський код додання продукту мовою TypeScript:

```
'use client';

import { useState } from 'react';
import axios, { CancelTokenSource } from 'axios';
import { Box, Button, Alert, CircularProgress } from '@mui/material';
import {
  SubmitHandler,
  useForm,
  Controller,
  useFormState,
} from 'react-hook-form';

import GlobalInput from '@/components/ElementsAndBlocks/GlobalInput';

import { useProductsStore } from '@/store/productsStore';

import {
  addProduct,
  addProductUsingARobot,
} from '@/services/products/productsService';
import { addReport } from '@/services/reports/reportsService';

import { getQueryByNameFromUrl } from '@/helpers/locationHelpers';
import { generateReportBody } from '@/helpers/reportHelpers';
```

```
import { qSKeys } from '@/constants/qSKeys';

import { IAddProductReqBody } from '@/services/products/productsTypes';
import { IProduct } from '@/types/product';
import { TError } from '@/types/error';
import { lSKeys } from '@/constants/lSKeys';

interface IAddProductFormProps {
  popupState: any;
}

let source: CancelTokenSource;

const AddProductForm: React.FC<IAddProductFormProps> = ({ popupState }) => {
  const { getProductByIDAction, getAllProductsAction, allProducts } =
    useProductsStore();

  const { handleSubmit, control } = useForm<IAddProductReqBody>();
  const { errors } = useFormState({
    control,
  });

  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState('');
  const [product, setProduct] = useState<IProduct | null>(null);

  const onSubmit: SubmitHandler<IAddProductReqBody> = async ({
    productID,
    shelveID,
    width,
    height,
    length,
    productTitle,
    productDescription,
    productImgUrl,
```

```
}) => {
  try {
    if (isLoading) return;

    setIsLoading(true);
    setError('');
    setProduct(null);

    const robotIP = localStorage.getItem(lSKeys.robotIP) || '';

    source = axios.CancelToken.source();

    const res = await addProductUsingARobot(robotIP, shelveID, productID, {
      cancelToken: source.token,
    });

    const { data: product } = await addProduct({
      productID,
      shelveID,
      width,
      height,
      length,
      productTitle,
      productDescription,
      productImgUrl,
    });

    setProduct(product);

    const productSearchVal = getQueryByNameFromUrl(qSKeys.productSearch);

    if (product && productSearchVal)
      getProductByIDAction(Number(productSearchVal));
    else if (product && !productSearchVal) getAllProductsAction();
```

```
    popupState.close();

    const addProductReport = generateReportBody(
      `Add product with id = ${productID}`,
    );
    await addReport(addProductReport);
  } catch (err: TError) {
    const error = err?.message ? err.message : 'Get product error';

    setError(error);
  } finally {
    setIsLoading(isLoading);
  }
};

const submitProductFormBtnHandler = () => {
  if (isLoading) {
    // source.cancel('Cancel query');
    setIsLoading(false);
  }
};

const maxProductID = allProducts.reduce((maxID, product) => {
  return Math.max(maxID, product.productID);
}, 0);

return (
  <form onSubmit={handleSubmit(onSubmit)}>
    <Controller
      control={control}
      name='productID'
      rules={numberRules}
      defaultValue={maxProductID + 1}
      render={({ field: { onChange, value } }: any) => (
        <GlobalInput
```

```
      value={value}
      onChange={onChange}
      required
      label='Product id'
      type='number'
      placeholder='0000'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.productID?.message}
    />
  )}
/>
<Controller
  control={control}
  name='shelveID'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='Shelve id'
      type='number'
      placeholder='0000'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.shelveID?.message}
    />
  )}
/>
<Controller
  control={control}
  name='width'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
```

```
      onChange={onChange}
      required
      label='Product Width'
      type='number'
      placeholder='00'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.width?.message}
    />
  )}
/>
<Controller
  control={control}
  name='height'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      required
      label='Product Height'
      type='number'
      placeholder='00'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.height?.message}
    />
  )}
/>
<Controller
  control={control}
  name='length'
  rules={numberRules}
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
```

```
      required
      label='Product Length'
      type='number'
      placeholder='00'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.length?.message}
    />
  )}
/>
<Controller
  control={control}
  name='productTitle'
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      label='Product Title'
      placeholder='Title'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.productTitle?.message}
    />
  )}
/>
<Controller
  control={control}
  name='productDescription'
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      label='Product Description'
      placeholder='Description'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.productDescription?.message}
    />
```

```
  )}
/>
<Controller
  control={control}
  name='productImgUrl'
  render={({ field: { onChange, value } }: any) => (
    <GlobalInput
      value={value}
      onChange={onChange}
      label='Product Image Url'
      placeholder='Image Url'
      customStyle={{ marginBottom: 3 }}
      helperText={errors?.productImgUrl?.message}
    />
  )}
/>

{!isLoading && error && (
  <Alert severity='error' sx={{ marginBottom: 2 }}>
    {error}
  </Alert>
)}

<Box sx={{ width: '100%', display: 'flex', justifyContent: 'flex-end' }}>
  <Button
    variant='contained'
    size='medium'
    type='submit'
    sx={{ minWidth: '100px' }}
    onClick={submitProductFormBtnHandler}
  >
    {isLoading ? 'Cancel' : 'Add'}
    {isLoading && (
      <CircularProgress
        color='secondary'
```

```
            sx={{ marginLeft: 1 }}
            size={25}
           />
         )}
       </Button>
     </Box>
   </form>
  );
};

const numberRules = {
  required: {
    value: true,
    message: 'This field is required',
  },
  pattern: {
    value: /^\d+$/,
    message: 'Data is not correct',
  },
};

export default AddProductForm;
```

Серверний код додання продукту мовою TypeScript:

```
async addProduct(req: Request, res: Response) {
  try {
    const errors = validationResult(req);

    if (!errors.isEmpty())
      res.status(400).json({
        message: 'Data is not correct',
        errors: errors.array(),
      });

    const {
      productID,
```

```
  width,
  height,
  length,
  shelveID,
  productTitle,
  productDescription,
  productImgUrl,
} = req.body;

const savedProduct: IProduct | null | any =
  await productRepository.findOne<{
    productID: number;
  }>({
    productID,
  });

if (savedProduct?.productID)
  return res
    .status(404)
    .json({ message: `Product with id ${productID} is already exist` });

const savedShelve: IShelve | null | any = await shelveRepository.findOne<{
  shelveID: number;
}>({
  shelveID,
});

if (!savedShelve?.shelveID)
  return res
    .status(404)
    .json({ message: `Shelve with id ${shelveID} is not exist` });

const isProductFit = shelveService.getIsProductFit(
  savedShelve.shelveDimensions.width,
  savedShelve.shelveDimensions.height,
```

```
      savedShelve.shelveDimensions.length,
      savedShelve.percentBusyVolume,
      width,
      height,
      length,
    );

    if (!isProductFit)
      return res.status(400).json({
        message: `Shelve with id ${shelveID} is not enough space`,
      });

    const newProduct = await productRepository.create({
      productID,
      productDimensions: { width, height, length },
      shelveID,
      productTitle,
      productDescription,
      productImgUrl,
    });

    await productService.addProductOnShelve(
      shelveID,
      productID,
      width,
      height,
      length,
    );

    res.send(newProduct);
  } catch (err) {
    return res.status(500).json(err);
  }
}
```

Клієнтський код картки продукту з функціоналом його отримання або видалення мовою TypeScript:

```
'use client';

import React, { useEffect, useState } from 'react';
import axios, { CancelTokenSource } from 'axios';
import {
  Accordion,
  AccordionDetails,
  AccordionSummary,
  Alert,
  Box,
  Button,
  Card,
  CardContent,
  CircularProgress,
  Typography,
} from '@mui/material';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

import RemoveProductBtn from '@/pages/ProductsPage/components/RemoveProductBtn';

import { getProductUsingARobot } from '@/services/products/productsService';
import { addReport } from '@/services/reports/reportsService';

import { generateReportBody } from '@/helpers/reportHelpers';

import { lSKeys } from '@/constants/lSKeys';

import { IGetProductUsingARobotResBody } from '@/services/products/productsTypes';
import { TError } from '@/types/error';
import { IDimensions } from '@/types/sizeAndCoordinates';

interface IShelveCardProps {
  productID: number;
```

```
  shelveID: number;
  productDimensions: IDimensions;
  productTitle?: string;
  productDescription?: string;
  productImgUrl?: string;
  isOneRemovePopupOpened: boolean;
  setIsOneRemovePopupOpened: (isOneRemovePopupOpened: boolean) => void;
}

let source: CancelTokenSource;

const ProductCard: React.FC<IShelveCardProps> = ({
  productID,
  shelveID,
  productDimensions,
  productTitle,
  productDescription,
  productImgUrl,
  isOneRemovePopupOpened,
  setIsOneRemovePopupOpened,
}) => {
  const [isLoading, setIsLoading] = useState(false);
  const [product, setProduct] = useState<IGetProductUsingARobotResBody | null>(
    null,
  );
  const [error, setError] = useState('');

  const getProductUsingARobotHandler = async (productID: number) => {
    try {
      const robotIP = localStorage.getItem(lSKeys.robotIP);

      if (!robotIP) {
        setError('Robot was not found');
        return;
      }
```

```
    setIsLoading(true);

    source = axios.CancelToken.source();

    const { data } = await getProductUsingARobot(productID, robotIP, {
      cancelToken: source.token,
    });

    setProduct(data);

    const getProductWithRobotReport = generateReportBody(
      `Get product with id = ${productID} using a robot`,
    );
    await addReport(getProductWithRobotReport);
  } catch (err: TError) {
    const error = err?.message ? err.message : 'Robot error';

    setError(error);
  } finally {
    setIsLoading(false);
  }
};

const cancelGettingProductUsingARobot = () => {
  source.cancel('Cancel query');
  setIsLoading(false);
};

const productRobotHandler = () => {
  if (!isLoading) {
    getProductUsingARobotHandler(productID);
    return;
  }
  cancelGettingProductUsingARobot();
```

```
};

const isSuccess = Boolean(!isLoading && product?.productID && !error.length);

const [isShowSuccess, setIsShowSuccess] = useState(false);

const isDescriptionEmpty = Boolean(
  !productTitle && !productDescription && !productImgUrl,
);

useEffect(() => {
  if (isSuccess) {
    setIsShowSuccess(true);
    setTimeout(() => {
      setIsShowSuccess(false);
    }, 4000);
  }
}, [isSuccess]);

return (
  <Card key={shelveID}>
    <CardContent>
      <Typography component='h3' variant='h3'>
        Product id: {productID}
      </Typography>
      <Typography component='h3' variant='h3'>
        Shelve id: {shelveID}
      </Typography>
      {!isDescriptionEmpty && (
        <Accordion
          sx={{ boxShadow: 'none', paddingLeft: 0, paddingBottom: 0 }}
        >
          <AccordionSummary
            expandIcon={<ExpandMoreIcon />}
            aria-controls='panel1-content'
```

```
    id='panel1-header'
    sx={{
      padding: 0,
      fontWeight: 700,
      fontSize: 20,
    }}
  >
    More information
  </AccordionSummary>
  <AccordionDetails sx={{ padding: 0, paddingBottom: 3 }}>
    {productTitle && (
      <Typography component='h4' variant='h4'>
        {productTitle}
      </Typography>
    )}
    {productDescription && (
      <Typography component='p' variant='body1'>
        {productDescription}
      </Typography>
    )}
    {productImgUrl && (
      /* eslint-disable-next-line @next/next/no-img-element */
      <img
        src={`https://drive.google.com/thumbnail?id=${productImgUrl}`}
        alt={productTitle || 'product image'}
        style={{
          width: '100%',
          height: 'auto',
          borderRadius: 3,
          marginTop: 5,
        }}
      />
    )}
  </AccordionDetails>
</Accordion>
```

```
)}
<Box sx={{ marginTop: 1 }}>
  <Typography component='h4' variant='h4'>
    Product dimensions:
  </Typography>
  <Typography component='p' variant='body1'>
    width: {productDimensions?.width} sm;
    <br />
    length: {productDimensions?.length} sm;
    <br />
    height: {productDimensions?.height} sm;
  </Typography>
</Box>
<Box
  sx={{
    width: '100%',
    display: 'flex',
    justifyContent: 'flex-end',
    marginTop: 2,
    gap: 2,
  }}
>
  <RemoveProductBtn
    {...{
      isOneRemovePopupOpened,
      setIsOneRemovePopupOpened,
      productID,
      setError,
    }}
  />
  <Button
    variant='contained'
    size='medium'
    onClick={productRobotHandler}
  >
```

```
          {isLoading ? 'Cancel' : 'Get'}
          {isLoading && (
            <CircularProgress
              color='secondary'
              sx={{ marginLeft: 1 }}
              size={25}
            />
          )}
        </Button>
      </Box>
      {error && (
        <Alert sx={{ marginTop: 2 }} severity='error'>
          {error}
        </Alert>
      )}
      {isShowSuccess && (
        <Alert sx={{ marginTop: 2 }}>{product?.message}</Alert>
      )}
    </CardContent>
  </Card>
 );
};

export default ProductCard;

'use client';

import { useState } from 'react';
import {
 Popper,
 Button,
 Fade,
 Paper,
 Typography,
 Box,
```

```
  CircularProgress,
} from '@mui/material';
import PopupState, { bindToggle, bindPopper } from 'material-ui-popup-state';

import { useProductsStore } from '@/store/productsStore';

import { removeProductByID } from '@/services/products/productsService';

import { getQueryByNameFromUrl } from '@/helpers/locationHelpers';

import { qSKeys } from '@/constants/qSKeys';
import { basicTheme } from '@/theme/theme';

import { TError } from '@/types/error';
import { generateReportBody } from '@/helpers/reportHelpers';
import { addReport } from '@/services/reports/reportsService';

interface IRemoveProductBtnProps {
  isOneRemovePopupOpened: boolean;
  setIsOneRemovePopupOpened: (isOneRemovePopupOpened: boolean) => void;
  productID: number;
  setError: (error: string) => void;
}

const RemoveProductBtn: React.FC<IRemoveProductBtnProps> = ({
  isOneRemovePopupOpened,
  setIsOneRemovePopupOpened,
  productID,
  setError,
}) => {
  const { getProductByIDAction, getAllProductsAction } = useProductsStore();

  const [isLoading, setIsLoading] = useState(false);

  const removeProductHandler = async (popupState: any) => {
```

```
  try {
    setIsLoading(true);
    setError('');

    const { data: product } = await removeProductByID(productID);

    const productSearchVal = getQueryByNameFromUrl(qSKeys.productSearch);

    if (product && productSearchVal)
      getProductByIDAction(Number(productSearchVal));
    else if (product && !productSearchVal) getAllProductsAction();

    const removeProductReport = generateReportBody(
      `Remove product with id = ${productID}`,
    );
    await addReport(removeProductReport);
  } catch (err: TError) {
    const error = err?.message ? err.message : 'Remove product error';

    setError(error);
  } finally {
    popupState.close();
    setIsOneRemovePopupOpened(false);
    setIsLoading(false);
  }
};

return (
  <PopupState variant='popper' popupId='demo-popup-popper' disableAutoFocus>
    {(popupState) => (
      <div>
        <Button
          variant='contained'
          disabled={isOneRemovePopupOpened}
          {...bindToggle(popupState)}
```

```
>
  Remove
  {isLoading && (
    <CircularProgress
      color='secondary'
      sx={{ marginLeft: 1 }}
      size={25}
    />
  )}
</Button>
<Popper {...bindPopper(popupState)} transition placement='bottom-end'>
  {({ TransitionProps }) => {
    setIsOneRemovePopupOpened(popupState.isOpen);

    return (
      <Fade {...TransitionProps} timeout={350}>
        <Paper
          sx={{ marginTop: 1, marginBottom: 1, p: 2, minWidth: 450 }}
        >
          <Typography
            component='h4'
            variant='h4'
            color={basicTheme.warning}
          >
            Are you want to remove this product?
          </Typography>
          <Box sx={{ marginTop: 2, display: 'flex', gap: 2 }}>
            <Button
              sx={{ color: basicTheme.success }}
              onClick={() => popupState.close()}
            >
              No
            </Button>
            <Button
              sx={{ color: basicTheme.error }}
```

```
                onClick={() => removeProductHandler(popupState)}
              >
                Yes
              </Button>
            </Box>
          </Paper>
        </Fade>
      );
    }}
  </Popper>
 </div>
 )}
</PopupState>
);
};

export default RemoveProductBtn;
```

Серверний код видалення продукту мовою TypeScript:

```
async deleteProduct(req: Request, res: Response) {
 try {
  const productID = Number(req.params.id);

  const product = await productRepository.findOne<{ productID: number }>({
   productID,
  });

  const {
   productDimensions: { width, height, length },
  } = product;

  if (!product?.shelveID)
   return res
    .status(404)
    .json({ message: `Product with id ${productID} is not exist` });
```

```
    const productDeleteResult = await productRepository.deleteByID(productID);

    if (!productDeleteResult.deletedCount)
      return res.status(404).json({ message: `Product deletion error` });

    await productService.deleteProductWithShelve(
      product.shelveID,
      productID,
      width,
      height,
      length,
    );

    return res.send(productDeleteResult);
  } catch (err) {
    return res.status(500).json(err);
  }
}
```

Серверний код макету робота для емуляції отримання продукту мовою C:

```
void getProductHandler() {
        isLoadingGoods = true;

        String goodsIdStr = server.arg("id");
        int localGoodsId = goodsIdStr.toInt();

        while(true) {
                if(goodsId == localGoodsId) break;
                delay(500);
        }

        delay(3000);

        DynamicJsonDocument jsonDoc(256);

        jsonDoc["message"] = "Product on the place";
```

```
        jsonDoc[“productID”] = goodsId;

        String jsonResponse;
        serializeJson(jsonDoc, jsonResponse);

        server.sendHeader(“Access-Control-Allow-Origin”, “*”);
        server.sendHeader(“Access-Control-Allow-Methods”, “GET, PUT, PATCH, POST,
DELETE”);
        server.sendHeader(“Access-Control-Allow-Headers”, “Content-Type, Authorization”);
        server.send(200, “application/json”, jsonResponse);
        isLoadingGoods = false;

        goodsId = -1;
}
```

ДОДАТОК Б

Демонстраційний матеріал

Харківський національний університет радіоелектроники

Кафедра КІТАР

Кваліфікаційна робота на тему:

Розроблення автоматизованої системи віддаленого управління роботизованим сховищем на виробничому підприємстві

Виконав:
Студент групи АКТАКІТ-20-1
Александрович Даніель Павлович

Керівник:
Професор кафедри КІТАР
Новоселов Сергій Павлович

Постановка задачі

Мета роботи

Покращення методів керування логістикою на складах та підвищення рівня автоматизації і безпеки роботи складських приміщень на виробничих підприємствах

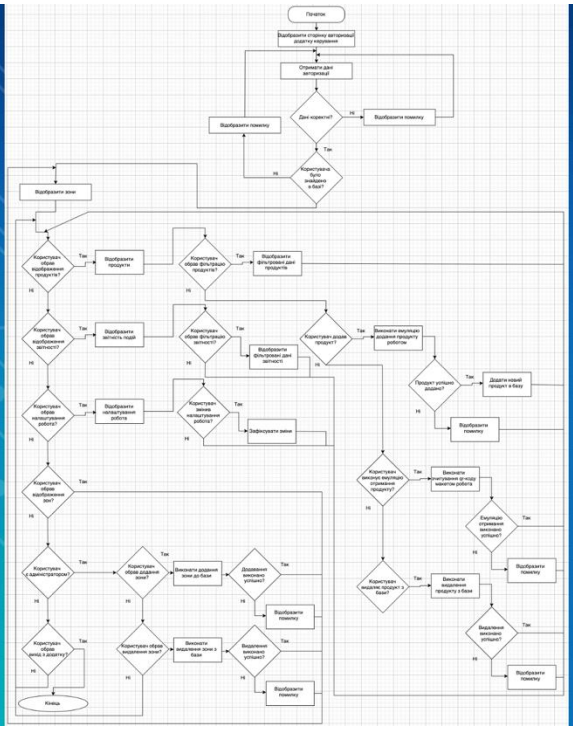
Для досягнення поставленої мети необхідно вирішити такі завдання:

- розробити структурну схему макету та алгоритм роботи системи;
- описати методи та технології взаємодії компонентів системи;
- провести підбір елементної бази та технологій реалізації компонентів макету системи;
- виконати реалізацію складових макету та їх поєднання;
- провести тестування розробленого макету.

Структурна схема макету та алгоритм роботи системи

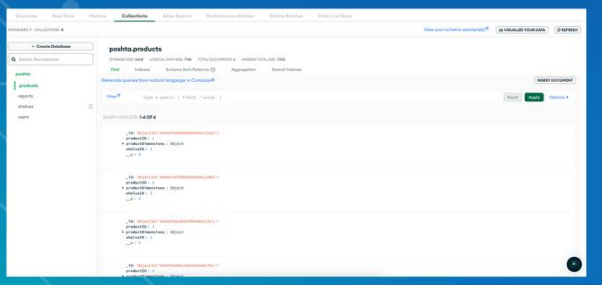
1. Додаток керування, що складається з двох під-додатків
2. Макет роботизованого пристрою на базі плати з доступом до будь-якої мережі
3. Макет сховища
4. База даних

Основна суть алгоритму роботи системи полягає у реакції бази даних або макету робота на певну подію: авторизація, додавання продукту, його отримання тощо.



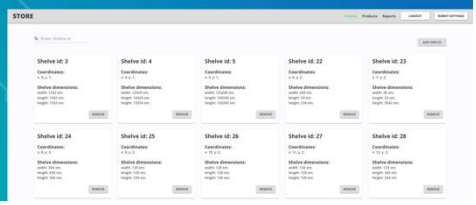
Елементна база та технології реалізації компонентів макету системи

В якості бази даних використано MongoDb, що була розгорнута на однойменному віддаленому хостингу.



Додаток керування реалізовано із використанням наступних технологій:

Клієнтський під-додаток:
- TypeScript, як мова програмування;
- Фреймворк Next;
- бібліотека керування станом Zustand.



Серверний під-додаток:
- TypeScript, як мова програмування;
- Фреймворк Express;
- бібліотека Mongoose для легкої взаємодії з базою даних MongoDb.

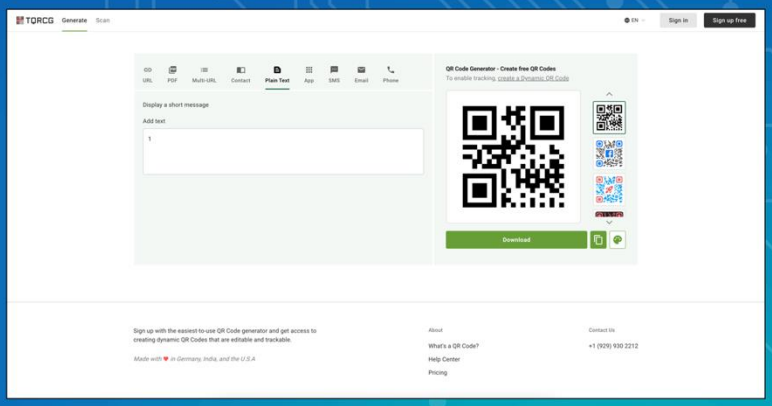
До складу макету робота входять: модуль esp-32cam, камера OV2640 та чотирьох колісна опорна платформа.

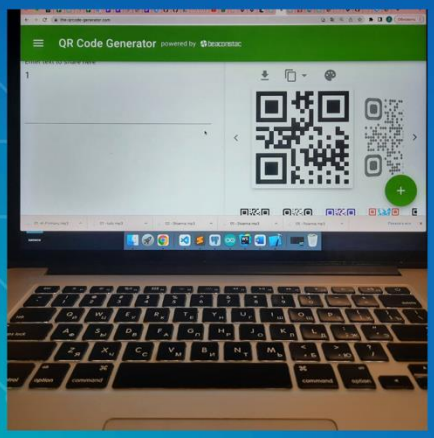
Камера OV2640 має матрицю на 2мп та кутом огляду 66° та підходить до модулів ESP32-CAM, ESP8266, Raspberry Pi, Arduino. Даної якості зображення достатньо для реалізації сканеру qr-коду

Має розміри 235 мм x 155 мм, просту конструкцію зручність в збиранні, має можливості розширення та дообладнання

Плата esp32-cam має wifi-модуль, слот для камери OV2640, а також може мати адаптер під роз'єм MicroUSB. Даний модуль є дешевим та багатофункціональним рішенням для безлічі завдань.

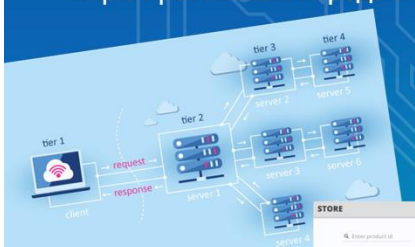
Для створення макету продукту в сховищі використано ноутбук з відкритим сервісом QR Code Generator. Даний ресурс надає можливість створення QR-коду з різних видів даних.

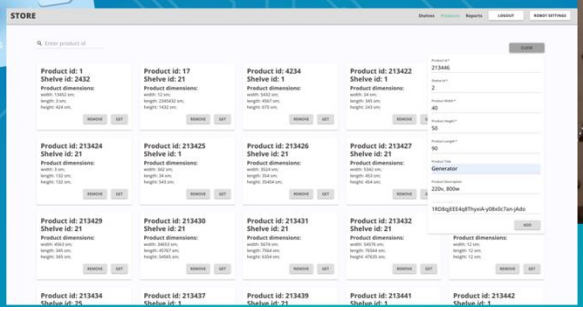


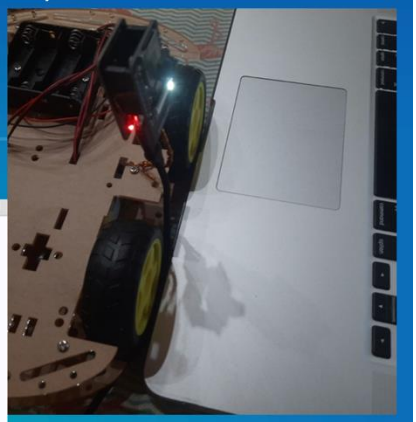


Методи та технології взаємодії компонентів системи

Взаємодія додатку керування та макету робота відбувається за клієнт-серверною архітектурою із використанням протоколу http.
http є протоколом передачі гіпертексту, тобто тексту-посилання на щось







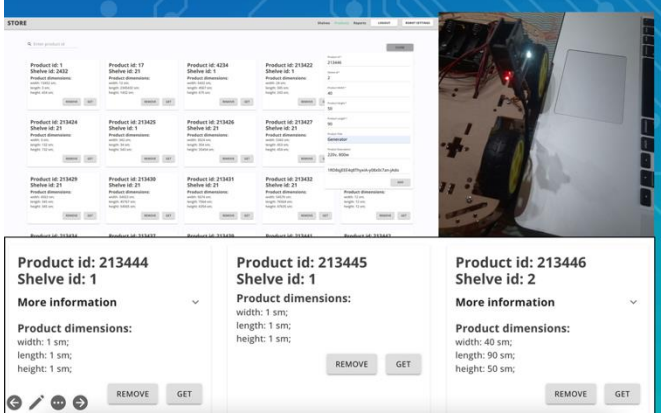
Взаємодія макету робота та сховища відбувається завдяки зчитування qr-коду камерою

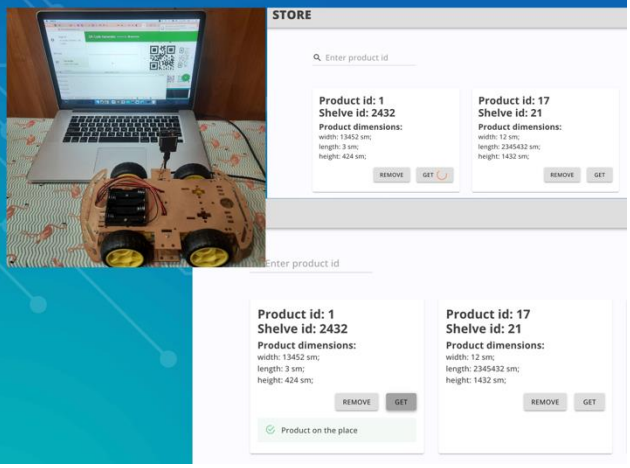
Базу даних розгорнуто на віддаленому сервісі, тому додаток керування взаємодіє з базою завдяки підключенню віддаленому підключенню при запуску сервера

```
await mongoose.connect( uri: `${process.env.DB_URL}`);
// mongodb+srv://danielaleksandrovych:Dan@cluster0.13sazaq.mongodb.net/poshta
```

Тестування основних функцій системи

Основними функціями системи є ті функції, у яких відбувається пряма взаємодія з макетом робота, тобто додання або отримання продукту.





Висновки

Під час виконання роботи, було поставлено та виконано ряд наступних завдань:

- розроблено структурну схему макету та алгоритм роботи системи;
- описано методи та технології взаємодії компонентів системи;
- проведено підбір елементної бази та технологій реалізації компонентів макету системи;
- виконано реалізацію складових макету та їх поєднання;
- проведено тестування розробленого макету.

Також було виконано аналіз сучасних складських роботів та складських приміщень. У рамках аналізу була проведена класифікація складських приміщень та складських роботів.

Проаналізовано існуючи види програмних додатків, технології для розробки та необхідність середовищ розробки. Також було виконано аналіз клієнт-серверної архітектури, як основи багатьох додатків.

Розроблено програмний додаток з інтерфейсом керування та отримання даних звітності. Також розроблено структуру бази даних, яка зберігається у віддаленому сервісі.

Для повноцінного функціонування було використано макет складського роботу - модуль esp32-cam з камерою та чотириколісною опорою та використано онлайн-сервіс генерації QR коду.

	Позначення	Найменування	Дод. Відомості
		Текстові документи	
1	ГЮІК.421455.001 ПЗ	Пояснювальна записка	58 с
		Графічні документи	
2		Демонстраційний матеріал	у форматі
			11 с. *.ppt

					ГЮІК.421455.001ВД	
Змін.	Арк.	Номер докум.	Підп.	Дата		
Розроб.		Александрович Д.П.			Розроблення автоматизованої системи віддаленого управління роботизованим сховищем на виробничому підприємстві	Аркуш: — Аркушів: 1
Перевір.		Новоселов С.П.				
Н.контр.		Стародубцев М.Г.				ХНУРЕ Кафедра КІТАМ
Затв.		Невлюдов І.Ш.				