ДОДАТОК А

Реєстрація користувача в системі

```
import { useQueryClient } from "@tanstack/react-query";
import { useNavigate } from "react-router-dom";
import { jwtDecode } from "jwt-decode";

const BASE_URL = import.meta.env.VITE_API_BASE_URL;

export function useAuth() {
  const queryClient = useQueryClient();
  const navigate = useNavigate();
  let refreshPromise = null;

  const refreshAccessToken = async () => {
    if (refreshPromise) {
      return refreshPromise;
    }

    refreshPromise = (async () => {
      const refreshToken = localStorage.getItem("refreshToken");
      const accessToken = localStorage.getItem("accessToken");

      if (!refreshToken) {
        logout();
      }

      const response = await fetch(`${BASE_URL}/tokens/refresh`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ accessToken, refreshToken }),
      });

      if (!response.ok) {
        logout();
      }

      const data = await response.json();
      localStorage.setItem("refreshToken", data.refreshToken);
      localStorage.setItem("accessToken", data.accessToken);

      refreshPromise = null; // Reset after completion
      return data;
    })();

    return refreshPromise;
  };

  const decodeToken = (accessToken = localStorage.getItem("accessToken")) =>
{
    try {
      const decodedData = jwtDecode(accessToken);
      const cleanedData = {
        name: decodedData[
```

```
          "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
        ],
        id: decodedData[

"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"
        ],
        email:
          decodedData[

"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
          ],
        role: decodedData[
          "http://schemas.microsoft.com/ws/2008/06/identity/claims/role"
        ],
        exp: decodedData.exp,
      };
      return cleanedData;
    } catch (err) {
      console.log("Invalid token:", err);
      // throw new Error(`Invalid token: ${err.message}`);
      return null;
    }
  };
  const getCurrentUser = async () => {
    const refreshToken = localStorage.getItem("refreshToken");
    let accessToken = localStorage.getItem("accessToken");
    if (!accessToken || !refreshToken) {
      return null;
    }

    let user = decodeToken(accessToken);
    if (!user) return null;
    const currentTime = Math.floor(Date.now() / 1000);
    if (user.exp < currentTime) {
      try {
        const refreshedData = await refreshAccessToken();
        accessToken = refreshedData.accessToken;
        user = decodeToken(accessToken);
      } catch (error) {
        return null;
      }
    }

    return user;
  };
  const logout = () => {
    localStorage.removeItem("refreshToken");
    localStorage.removeItem("accessToken");
  };
  return {
    getCurrentUser,
    refreshAccessToken,
    decodeToken,
    logout,
  };
}
```

ДОДАТОК Б

Архітектура файлового дерева

Folders.jsx

```
import { useMediaQuery } from "react-responsive";
import Modal from "../../ui/Modal";
import { OpenFoldersProvider } from "../../context/OpenFoldersContext";
import { HiOutlineFolderOpen } from "react-icons/hi";
import styled from "styled-components";
import Button from "../../styles/Button";
import FolderTree from "./FolderTree";
import { EditedFolderProvider } from "../../context/EditedFolderContext";
import { EditedFileProvider } from "../../context/EditedFileContext";
const FoldersContainer = styled.div`
  height: 52vh;
  background-color: var(--gray-600);
  border-radius: var(--border-radius-l);
  padding-top: 1rem;
  font-size: 17px;
  overflow: auto;
  line-height: 2rem;
  @media (max-width: 768px) {
    width: 100vw;
    height: 100vh;
    border-radius: 0%;
    padding-top: 4rem;
  }
`;
const Title = styled.p`
  color: var(--white);
  font-weight: bolder;
  margin-bottom: 0.8rem;
  letter-spacing: 0.5px;
  font-size: 18px;
  padding-left: 1.2rem;
`;
const FilterButton = styled(Button)`
  justify-self: start;
  display: flex;
  align-items: center;
  gap: 0.3rem;
  font-size: 16px;
  padding: 0%;
  cursor: pointer;
  color: var(--yellow);
`;

function Folders({ rootFolder, readOnly = true }) {
  const isMobile = useMediaQuery({ query: "(max-width: 768px)" });
  if (isMobile) {
    return (
      <EditedFolderProvider>
        <EditedFileProvider>
          <Modal>
```

```
                <Modal.Open opens="folders">
                  <FilterButton variation="icon">
                    <HiOutlineFolderOpen /> Files
                  </FilterButton>
                </Modal.Open>
                <Modal.Window name="folders">
                  <OpenFoldersProvider>
                    <FoldersContainer>
                      <Title>Files</Title>
                      <FolderTree folder={rootFolder} readOnly={readOnly}
/>
                    </FoldersContainer>
                  </OpenFoldersProvider>
                </Modal.Window>
              </Modal>
            </EditedFileProvider>
          </EditedFolderProvider>
        );
      }
      return (
        <EditedFolderProvider>
          <EditedFileProvider>
            <OpenFoldersProvider>
              <FoldersContainer>
                <Title>Files</Title>
                <FolderTree folder={rootFolder} readOnly={readOnly} />
              </FoldersContainer>
            </OpenFoldersProvider>
          </EditedFileProvider>
        </EditedFolderProvider>
      );
    }
    export default Folders;
```

FolderTree.jsx

```
    import styled from "styled-components";
    import { useOpenFolders } from "../../context/OpenFoldersContext.jsx";
    import File from "./File.jsx";
    import Folder from "./Folder.jsx";
    import { useEditedFolderContext } from
"../../context/EditedFolderContext.jsx";
    import { useEffect, useState } from "react";
    import { useEditedFileContext } from
"../../context/EditedFileContext.jsx";

    const FolderTreeContainer = styled.div`
      display: flex;
      flex-direction: column;
      min-width: 100%;
    `;
    const OpenedFolderContent = styled.div`
      display: flex;
      flex-direction: column;
      margin-left: 1.5rem;
      border-left: 1px solid var(--gray-500);
    `;
```

```
function FolderTree({ folder, readOnly = true }) {
  const { openFolders } = useOpenFolders();
  const isOpen = openFolders[folder.id] || false;
  const { setEditingFolderId } = useEditedFolderContext();
  const [newFolderId, setNewFolderId] = useState(null);
  const { setEditingFileId } = useEditedFileContext();
  const [newFileId, setNewFileId] = useState(null);

  useEffect(() => {
    if (newFolderId) {
      setEditingFolderId(newFolderId);
      setNewFolderId(null);
    }
  }, [newFolderId, setEditingFolderId]);

  useEffect(() => {
    if (newFileId) {
      setEditingFileId(newFileId);
      setNewFileId(null);
    }
  }, [newFileId, setEditingFileId]);

  const handleFolderCreated = (newFolderId) => {
    setNewFolderId(newFolderId);
  };

  const handleFileCreated = (newFileId) => {
    setNewFileId(newFileId);
  };
  return (
    <FolderTreeContainer>
      <Folder
        folder={folder}
        readOnly={readOnly}
        onFolderCreated={handleFolderCreated}
        onFileCreated={handleFileCreated}
      />
      {isOpen && (
        <OpenedFolderContent>
          {folder.items.map((item) =>
            item.type === 0 ? (
              <FolderTree          key={item.id}          folder={item}
readOnly={readOnly} />
            ) : (
              <File key={item.id} file={item} readOnly={readOnly} />
            )
          )}
        </OpenedFolderContent>
      )}
    </FolderTreeContainer>
  );
}
export default FolderTree;
```

Folder.jsx

```
import { HiChevronDown, HiChevronRight, HiOutlineTrash } from "react-
icons/hi";
import { useOpenFolders } from "../../context/OpenFoldersContext";
import styled from "styled-components";
import { DropdownMenu } from "radix-ui";
import {
  HiOutlineDocumentPlus,
  HiOutlineEllipsisVertical,
  HiOutlineFolderPlus,
  HiOutlinePencilSquare,
} from "react-icons/hi2";
import {
  Operations,
  StyledContent,
  StyledItem,
  StyledToggle,
} from "../../styles/DropdownMenu";
import { useAsset } from "../assets/useAsset";
import { useEffect, useRef, useState } from "react";
import { useRenameFolder } from "./useRenameFolder";
import findParentFolder from "../../utils/findParentFolder";
import toast from "react-hot-toast";
import { useCreateFolder } from "./useCreateFolder";
import { useDeleteFolder } from "./useDeleteFolder";
import containsPrimaryCodeFile from
"../../utils/containsPrimaryCodeFile";
import { useEditedFolderContext } from
"../../context/EditedFolderContext";
import { useCreateFile } from "./useCreateFile";

const StyledFolder = styled.div`
  min-height: 2rem;
  cursor: pointer;
  display: flex;
  align-items: center;
  gap: 2px;
  padding: 3px 0px;
  padding-left: 1rem;

  & svg {
    color: var(--white);
  }
  &:hover {
    background-color: var(--gray-400);
  }
`;
const Name = styled.input`
  pointer-events: ${(props) => (props.isDisabled ? "none" : "auto")};
  width: 100%;
  background-color: transparent;
  border: none;
`;

function Folder({ folder, readOnly = true, onFolderCreated,
onFileCreated }) {
  const { openFolders, toggleFolder } = useOpenFolders();
  const { asset } = useAsset();
```

```
        const { renameFolder, isPending } = useRenameFolder();
        const { createFolder } = useCreateFolder();
        const { deleteFolder } = useDeleteFolder();
        const { createFile } = useCreateFile();

        const isOpen = openFolders[folder.id] || false;
        const     {     editingFolderId,     setEditingFolderId     }     =
useEditedFolderContext();
        const [name, setName] = useState(folder.name);
        const nameInputRef = useRef(null);
        const hasPrimaryCodeFile = containsPrimaryCodeFile(
          folder,
          asset.primaryCodeFile.id
        );

        useEffect(() => {
          if (editingFolderId === folder.id) {
            setTimeout(() => {
              nameInputRef.current?.focus();
            }, 10);
          }
        }, [editingFolderId, folder.id]);

        const handleRename = () => {
          if (name.trim() && name !== folder.name) {
            const parent = findParentFolder(asset.rootFolder, folder.id);
            if (parent) {
              const isDuplicate = parent.items.some(
                (f) =>
                  f.id   !==   folder.id   &&   f.name.toLowerCase()   ===
name.toLowerCase()
              );
              if (isDuplicate) {
                toast.error("A  file  or  folder  with  such  name  already
exists");
                setName(folder.name);
                return;
              }
            }
            renameFolder({
              id: folder.id,
              name,
              parentId: parent?.id || null,
            });
          }
          setEditingFolderId(null);
        };

        const handleCreateFolder = () => {
          createFolder(
            { name: "", parentId: folder.id },
            {
              onSuccess: (newFolder) => {
                onFolderCreated(newFolder.id);
              },
            }
          );
```

```
        };

        const handleDelete = () => {
          deleteFolder({ id: folder.id });
        };
        const handleCreateFile = () => {
          createFile(
            { name: "", text: "// Write your code here", parentId: folder.id
},
            {
              onSuccess: (newFile) => {
                onFileCreated(newFile.id);
              },
            }
          );
        };
        return (
          <StyledFolder
            onClick={()    =>    editingFolderId    !==    folder.id    &&
toggleFolder(folder.id)}
          >
            {isOpen ? <HiChevronDown /> : <HiChevronRight />}

            <Name
              ref={editingFolderId === folder.id ? nameInputRef : null}
              value={name}
              onChange={(e) => setName(e.target.value)}
              onBlur={handleRename}
              onKeyDown={(e) => {
                if (e.key === "Enter") {
                  handleRename();
                  e.target.blur();
                }
              }}
              isDisabled={editingFolderId !== folder.id || isPending}
            />

            {!readOnly && (
              <Operations isActive={isOpen}>
                <DropdownMenu.Root>
                  <StyledToggle>
                    <HiOutlineEllipsisVertical />
                  </StyledToggle>

                  <StyledContent>
                    <StyledItem
                      onClick={(e) => {
                        e.stopPropagation();
                        handleCreateFile();
                      }}
                    >
                      <HiOutlineDocumentPlus /> New file
                    </StyledItem>
                    <StyledItem
                      onClick={(e) => {
                        e.stopPropagation();
                        handleCreateFolder();
```

```
                }}
              >
                <HiOutlineFolderPlus /> New folder
              </StyledItem>
              <StyledItem
                onClick={(e) => {
                  e.stopPropagation();
                  setEditingFolderId(folder.id);
                }}
              >
                <HiOutlinePencilSquare />
                Rename
              </StyledItem>
              {!hasPrimaryCodeFile && (
                <StyledItem
                  onClick={(e) => {
                    e.stopPropagation();
                    handleDelete();
                  }}
                >
                  <HiOutlineTrash /> Delete
                </StyledItem>
              )}
            </StyledContent>
          </DropdownMenu.Root>
        </Operations>
      )}
    </StyledFolder>
  );
}

export default Folder;
```

File.jsx

```
import styled from "styled-components";
import { useActiveFile } from "../../context/ActiveFileContext";
import { HiOutlineStar, HiOutlineTrash } from "react-icons/hi";
import {
  HiMiniStar,
  HiOutlineEllipsisVertical,
  HiOutlinePencilSquare,
} from "react-icons/hi2";
import { DropdownMenu } from "radix-ui";
import { useAsset } from "../assets/useAsset";
import {
  Operations,
  StyledContent,
  StyledItem,
  StyledToggle,
} from "../../styles/DropdownMenu";
import { useRenameFile } from "./useRenameFile";
import { useEffect, useRef, useState } from "react";
import findParentFolder from "../../utils/findParentFolder";
import toast from "react-hot-toast";
import { detectLanguage } from "../../utils/detectLanguage";
import { useEditedFileContext } from "../../context/EditedFileContext";
```

```
import { useDeleteFile } from "./useDeleteFile";
import { useEditAsset } from "../assets/useEditAsset";
import { useMakePrimaryCodeFile } from "./useMakePrimaryCodeFile";
import { useLocation } from "react-router-dom";

const StyledFile = styled.div`
  min-height: 2rem;
  display: flex;
  align-items: center;
  padding: 3px 0px;
  padding-left: 1rem;
  background-color: ${({ isActive }) =>
    isActive ? "var(--gray-500)" : "transparent"};
  cursor: pointer;
  &:hover {
    background-color: var(--gray-500);
  }
`;
const Name = styled.input`
  width: 100%;
  background-color: transparent;
  border: none;
  pointer-events: ${(props) => (props.isDisabled ? "none" : "auto")};
  &:disabled {
    background-color: transparent;
    cursor: pointer;
    color: var(--gray-200);
  }
`;
const Star = styled(HiMiniStar)`
  color: var(--yellow);
`;

function File({ file, readOnly = true }) {
  const { activeFile, setActiveFile } = useActiveFile();
  const { asset } = useAsset();
  const { makePrimaryCodeFile, isPending: isChangingFile } =
    useMakePrimaryCodeFile();
  const { renameFile } = useRenameFile();
  const { deleteFile, isPending: isDeleting } = useDeleteFile();
  const { editingFileId, setEditingFileId } = useEditedFileContext();
  const [name, setName] = useState(file.name);
  const nameInputRef = useRef(null);
  const location = useLocation();
  const isProject = location.pathname.startsWith("/project");

  useEffect(() => {
    if (editingFileId === file.id) {
      setTimeout(() => {
        if (nameInputRef.current) {
          nameInputRef.current.focus();
        }
      }, 10);
    }
  }, [editingFileId, file.id]);
  useEffect(() => {
    let toastId = null;
```

```
          if (isChangingFile || isDeleting) {
            toastId = toast.loading("Applying changes...");
          }

          return () => {
            if (toastId) {
              toast.dismiss(toastId);
            }
          };
        }, [isChangingFile, isDeleting]);

        const handleRename = () => {
          if (name.trim() && name !== file.name) {
            const extensionRegex = /\.[0-9a-z]+$/i;
            if (!extensionRegex.test(name)) {
              toast.error("File must have an extension");
              setName(file.name);
              return;
            }
            const currentExtension = file.name.match(extensionRegex)?.[0] ||
"";
            const newExtension = name.match(extensionRegex)?.[0] || "";
            let language = file.language;
            if (newExtension !== currentExtension) {
              language = detectLanguage(newExtension);
              if (language === "Unknown") {
                toast.error("Language is not supported");
                setName(file.name);
                return;
              }
            }
            const parent = findParentFolder(asset.rootFolder, file.id);
            if (parent) {
              const isDuplicate = parent.items.some(
                (f) => f.id !== file.id && f.name.toLowerCase() ===
name.toLowerCase()
              );

              if (isDuplicate) {
                toast.error("A file or folder with such name already
exists");
                setName(file.name);
                return;
              }
            }
            renameFile({
              id: file.id,
              text: file.text,
              language,
              name,
              parentId: parent.id,
            });
          }
          setEditingFileId(null);
        };
```

```
        const handleDelete = () => {
          deleteFile({ id: file.id });
        };

        const handlePrimaryCodeFile = () => {
          makePrimaryCodeFile({
            id: asset.id,
            description: asset.description,
            name: asset.name,
            tagsIds: asset.tags.map((tag) => tag.id),
            assetType: asset.assetType,
            language: file.language,
            rootFolderId: asset.rootFolder.id,
            primaryCodeFileId: file.id,
          });
        };

        return (
          <StyledFile
            onClick={(e) => {
              setActiveFile(file);
            }}
            isActive={activeFile?.id === file.id}
          >
            <Name
              ref={editingFileId === file.id ? nameInputRef : null}
              value={name}
              onChange={(e) => setName(e.target.value)}
              onBlur={handleRename}
              onKeyDown={(e) => {
                if (e.key === "Enter") {
                  handleRename();
                  e.target.blur();
                }
              }}
              isDisabled={editingFileId !== file.id || isDeleting ||
isChangingFile}
            />
            {!readOnly && (
              <Operations isActive={activeFile?.name === file.name}>
                {!isProject &&
                  (file.id === asset.primaryCodeFile.id ? (
                    <Star />
                  ) : (
                    <HiOutlineStar
                      onClick={(e) => {
                        e.stopPropagation();
                        handlePrimaryCodeFile();
                      }}
                    />
                  ))}
                <DropdownMenu.Root>
                  <StyledToggle>
                    <HiOutlineEllipsisVertical />
                  </StyledToggle>

                  <StyledContent>
```

```
              <StyledItem onClick={() => setEditingFileId(file.id)}>
                <HiOutlinePencilSquare /> Rename
              </StyledItem>
              {file.id !== asset.primaryCodeFile.id && (
                <StyledItem
                  onClick={(e) => {
                    e.stopPropagation();
                    handleDelete();
                  }}
                >
                  <HiOutlineTrash /> Delete
                </StyledItem>
              )}
            </StyledContent>
          </DropdownMenu.Root>
        </Operations>
      )}
    </StyledFile>
  );
}

export default File;
```

ДОДАТОК В

Тестування модулю створення кодового фрагмента

AddAssetForm.test.jsx

```
import { describe, it, expect, vi, beforeEach } from "vitest";
import { render, screen, fireEvent } from "@testing-library/react";
import AddAssetForm from "./AddAssetForm";
import toast from "react-hot-toast";

// Mock the toast module
vi.mock("react-hot-toast", () => ({
  default: {
    error: vi.fn(),
  },
}));

// Create mock function for createAsset
const mockCreateAsset = vi.fn();

// Setup our default mocks - this approach keeps mocks consistent
vi.mock("./useCreateAsset", () => ({
  useCreateAsset: () => ({
    createAsset: mockCreateAsset,
    isPending: false,
  }),
}));

// Default user role mock - using a getter function to allow changes
let currentUserRole = ["User"];
vi.mock("../authentication/useUser", () => ({
  useUser: () => ({
    user: {
      name: "John",
      get role() {
        return currentUserRole;
      },
    },
  }),
}));

// Mock utility function
vi.mock(
  "./utils",
  () => ({
    detectExtension: (lang) => {
      const extensions = {
        Javascript: ".js",
        Python: ".py",
        Csharp: ".cs",
      };
      return extensions[lang] || ".js";
    },
  }),
  { virtual: true }
);
```

```
      describe("AddAssetForm", () => {
        beforeEach(() => {
          // Reset all mocks before each test
          vi.clearAllMocks();
          currentUserRole = ["User"]; // Reset user role to default
        });

        it("renders form elements correctly", () => {
          render(<AddAssetForm />);

          // Check if all form elements are present
          expect(screen.getByText(/title:/i)).toBeInTheDocument();
          expect(screen.getByPlaceholderText(/provide                    a
name/i)).toBeInTheDocument();
          expect(screen.getByText(/language:/i)).toBeInTheDocument();
          expect(screen.getByText(/javascript/i)).toBeInTheDocument();
          expect(screen.getByRole("button",      {      name:      /create/i
})).toBeInTheDocument();
        });

        it("shows error toast if asset name is empty", () => {
          render(<AddAssetForm />);

          // Click create button without entering a name
          fireEvent.click(screen.getByRole("button", { name: /create/i }));

          // Check if error toast was called with the right message
          expect(toast.error).toHaveBeenCalledWith("Asset   must   have   a
title!");
          expect(mockCreateAsset).not.toHaveBeenCalled();
        });

        it("updates name input when user types", () => {
          render(<AddAssetForm />);

          const nameInput = screen.getByPlaceholderText(/provide a name/i);
          fireEvent.change(nameInput, { target: { value: "My New Asset" } });

          expect(nameInput.value).toBe("My New Asset");
        });

        it("changes language when selecting from dropdown", () => {
          render(<AddAssetForm />);

          // Open the dropdown
          fireEvent.click(screen.getByText(/javascript/i));

          // Select Python
          fireEvent.click(screen.getByText(/python/i));

          // Verify the value has changed
          expect(screen.getByText(/python/i)).toBeInTheDocument();
        });

        it("submits correct asset data for regular user", () => {
          render(<AddAssetForm />);
```

```
        // Enter asset name
        const nameInput = screen.getByPlaceholderText(/provide a name/i);
        fireEvent.change(nameInput, { target: { value: "Test Asset" } });

        // Click create button
        fireEvent.click(screen.getByRole("button", { name: /create/i }));

        // Check if createAsset was called with correct parameters
        expect(mockCreateAsset).toHaveBeenCalledWith({
          name: "Test Asset",
          assetType: 1, // Regular user gets assetType 1
          language: "Javascript", // Default is Javascript
          rootFolderName: "Test_Asset",
          primaryCodeFileName: "primary.js",
        });
      });

      it("submits correct asset data for enterprise user", () => {
        // Update user role to Enterprise for this test
        currentUserRole = ["Enterprise"];

        render(<AddAssetForm />);

        // Enter asset name
        const nameInput = screen.getByPlaceholderText(/provide a name/i);
        fireEvent.change(nameInput, { target: { value: "Enterprise Asset"
} });

        // Change language to Python
        fireEvent.click(screen.getByText(/javascript/i));
        fireEvent.click(screen.getByText(/python/i));
    // Click create button
        fireEvent.click(screen.getByRole("button", { name: /create/i }));

        // Check if createAsset was called with correct parameters
        expect(mockCreateAsset).toHaveBeenCalledWith({
          name: "Enterprise Asset",
          assetType: 2, // Enterprise user gets assetType 2
          language: "Python", // Changed to Python
          rootFolderName: "Enterprise_Asset",
          primaryCodeFileName: "primary.py",
        });
      });
    });
```

ДОДАТОК Г

Апробація результатів роботи

AI-powered Software Project Generation from Natural Language Input Using Pre-stored Code Assets in Vector Databases

Serhii Shchoholiev[1,†], Polina Pavlenko[1,*,†], Mariya Shirokopetleva[1,†] and Volodymyr Kobziev[1,†]

[1] *Kharkiv National University of Radio Electronics, Nauky Ave 14 61166 Kharkiv, Ukraine*

Abstract

The thesis addresses the challenge of efficiently generating software projects from natural language input. By integrating AI-powered solutions and leveraging pre-stored code assets in vector databases, the study examines methods for automating project creation. The results demonstrate the potential of AI to streamline development workflows, offering a more efficient approach to software development.

Keywords

code generation, artificial intelligence, code assets, vector databases

1. Introduction

As AI continues to gain traction among software development companies, its integration into various stages of the development process is becoming increasingly common. Automating the creation of source code, particularly for starter projects generation, has the potential to streamline and simplify workflows significantly.

This work examines the feasibility of using OpenAI to generate software projects by leveraging pre-stored code assets. Additionally, it explores existing AI technologies to assess their effectiveness in automating code assets generation and boosting overall productivity.

2. Analogue analysis

The research was conducted to analyze key analogues in the field of AI-powered code generation and management to identify their strengths and weaknesses.

CodeSnippets AI, the closest analogue to the proposed software, is an AI-driven tool for optimizing workflows and centralizing code management. It supports code generation, refactoring, debugging, and documentation, allowing developers to focus on complex tasks. However, limited team and role management features restrict its use in larger organizations, and access to personal/team snippets is confined without public sharing.

GitHub Copilot [1], developed by GitHub with OpenAI, is an AI-powered code completion tool providing context-aware suggestions in the IDE. By leveraging a vast dataset of publicly available code, it generates functions, suggests code, and offers real-time assistance. However, concerns remain about the quality, accuracy, and potential licensing issues of the generated code.

Information Systems and Technologies (IST-2024), November 26-28, 2024, Kharkiv, Ukraine

* Corresponding author.

† These authors contributed equally.

serhii.shchoholiev@nure.ua (S. Shchoholiev); polina.pavlenko@nure.ua (P. Pavlenko); marija.shirokopetleva@nure.ua (M. Shirokopetleva); volodymyr.kobziev@nure.ua (V. Kobziev)

0009-0007-2014-4828 (S. Shchoholiev); 0009-0003-1395-1630 (P. Pavlenko); 0000-0002-7472-6045 (M. Shirokopetleva); 0000-0002-8303-1595 (V. Kobziev)

3. Problem statement

In recent years, microservices have become a standard in software development. Despite their numerous advantages, they have amplified an existing challenge that many developers face — code duplication. Although various patterns have been developed over the years to address this issue, code duplication persists as a significant problem.

This research adopts a novel approach to addressing code duplication. Traditionally, developers strive to avoid code duplication to save development time — after all, if a solution already exists, repeating the work is inefficient. Instead of minimizing code duplication, this study embraces it while shifting the focus toward reducing overall development time.

4. Solution description

The proposed solution aims to leverage generative AI and vector databases to automate the process of software project generation from natural language inputs. By combining the capabilities of AI for understanding developer intents with the efficiency of vector databases for code asset retrieval, this approach seeks to reduce development time and improve productivity.

As with all AI-driven solutions, this approach requires data — specifically, code assets that are frequently reused within software projects. Company-wide services like file storage, identity management, and other common functionalities are used by numerous microservices, and the code to communicate with these services has already been written. By isolating these parts into distinct code assets, they can be stored in a database with vector representations for efficient retrieval. Leveraging the OpenAI embedding model 'text-embedding-3-small,' the code is converted into vector [2] form and stored in a vector database, enabling efficient search.

The next step enables developers to interact with the code asset database using natural language queries. We use a flagship generative AI model from OpenAI — gpt-4o-2024-08-06, which supports fine-tuning on proprietary datasets. Fine-tuning on metadata associated with code assets enhances the accuracy of asset selection. Through prompt engineering, including few-shot techniques [3] and function-calling, the model translates a developer's request into search queries against the vector database. Once retrieved, relevant code assets are verified with the developer, and a starter project is generated and compiled to ensure correctness.

5. Conclusions

This study demonstrates the potential of integrating AI-powered generative models with vector databases to automate software project generation from natural language inputs. By leveraging pre-stored code assets, this approach reduces project setup time, streamlines workflows, and enhances productivity. The solution enables efficient retrieval of reusable code components through natural language interaction, highlighting how AI can transform software development by reducing redundant tasks and allowing developers to focus on creative aspects.

References

[1] Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. (2023). doi:10.48550/arXiv.2304.10778

[2] Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Yuan, S., Tar, C., Sung, Y., Strope, B., & Kurzweil, R. Universal Sentence Encoder. (2018). doi:10.48550/arXiv.1803.11175

[3] Brown, Tom B., et al. Language Models are Few-Shot Learners. (2020). doi:10.48550/arXiv.2005.14165

ДОДАТОК Д

Виставка технічної творчості молоді

17. Програмна система для зберігання, спільного доступу та перевірки якості програмного коду

Автори: ***Щоголєв Сергій Андрійович, Білодід Михайло Олександрович, Павленко Поліна Олексіївна,*** ст. гр. ПЗПІ-21-4, ХНУРЕ.
Наукові керівники: Широкопетлєва М.С., ст. викл., Кобзєв В.Г., доц., каф. ПІ, ХНУРЕ.

Програмна система для зберігання фрагментів програмного коду та управління ними має широкий спектр застосування: її можна використовувати для організації спільного доступу до коду та його повторного використання, перевірки якості коду, контролю за компіляцією, генерації проєктів на підставі фрагментів коду.

Для оцінки якості програмного коду в системі використовується штучний інтелект сервісу OpenAI API, який аналізує код на відповідність принципам SOLID та DRY. Це сприяє підвищенню модульності, гнучкості та зрозумілості програмних рішень.

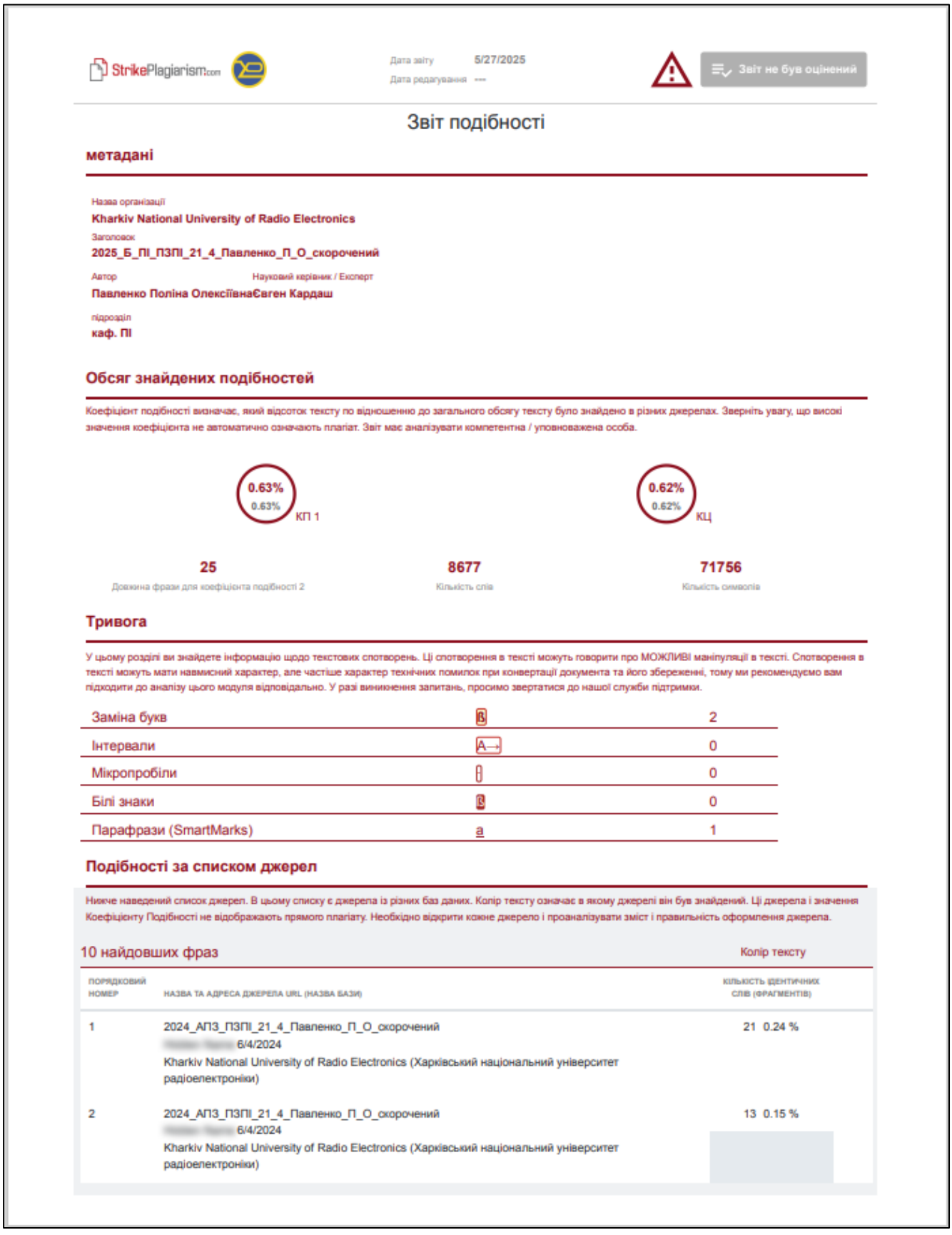
Для швидкого та ефективного доступу до даних система застосовує векторне представлення коду, перетворюючи текст програм на вектори. Такий підхід дозволяє спростити обробку коду, полегшити його аналіз та покращити роботу алгоритмів перевірки та пошуку фрагментів схожого коду для уникнення дублювання.

Окрім цього, система підтримує можливість спільного доступу до фрагментів коду всередині компанії. Це надає розробникам інструменти для повторного використання існуючих рішень, що дозволяє значно пришвидшити процес розробки. Також передбачено механізм генерації нових проєктів на основі наявного коду компанії, що спрощує створення нових програмних продуктів та сприяє ефективному управлінню кодовою базою.

Запропоноване рішення значно спрощує процес розробки програмного забезпечення, сприяє покращенню якості коду та підвищує ефективність роботи команд розробників завдяки можливості створення проєктів з існуючих перевірених фрагментів коду.

ДОДАТОК Е

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 5/27/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ_21_4_Павленко_П_О_скорочений
Автор
Павленко Поліна Олексіївна
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.63% 0.63% КП 1

0.62% 0.62% КЦ

25	8677	71756
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

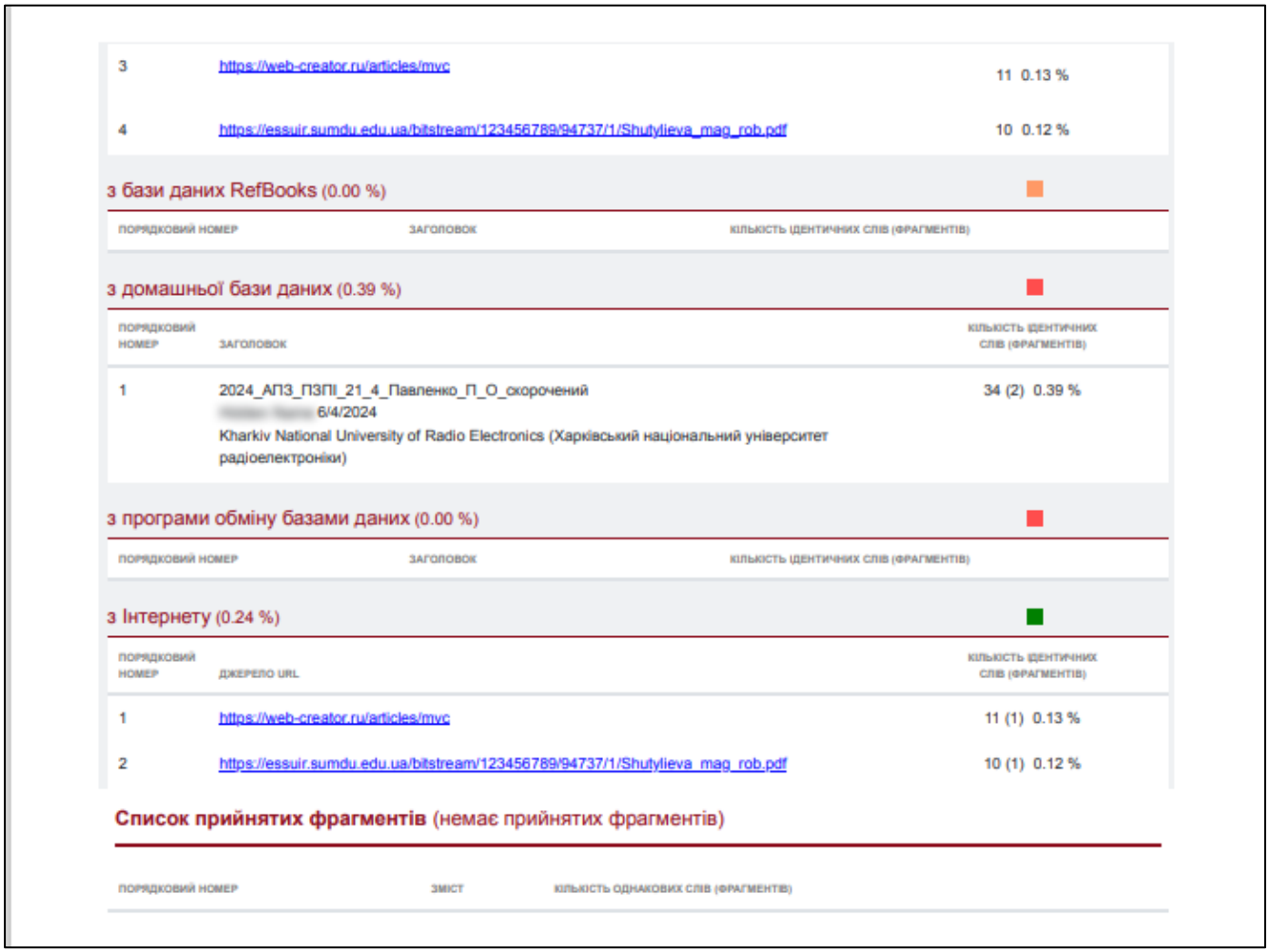
Заміна букв		2
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	1

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2024_АПЗ_ПЗПІ_21_4_Павленко_П_О_скорочений [redacted] 6/4/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	21 0.24 %
2	2024_АПЗ_ПЗПІ_21_4_Павленко_П_О_скорочений [redacted] 6/4/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	13 0.15 %



3	https://web-creator.ru/articles/mvc	11 0.13 %
4	https://essuir.sumdu.edu.ua/bitstream/123456789/94737/1/Shutylieva_mag_rob.pdf	10 0.12 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.39 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2024_АПЗ_ПЗПІ_21_4_Павленко_П_О_скорочений [redacted] 6/4/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	34 (2) 0.39 %

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (0.24 %)

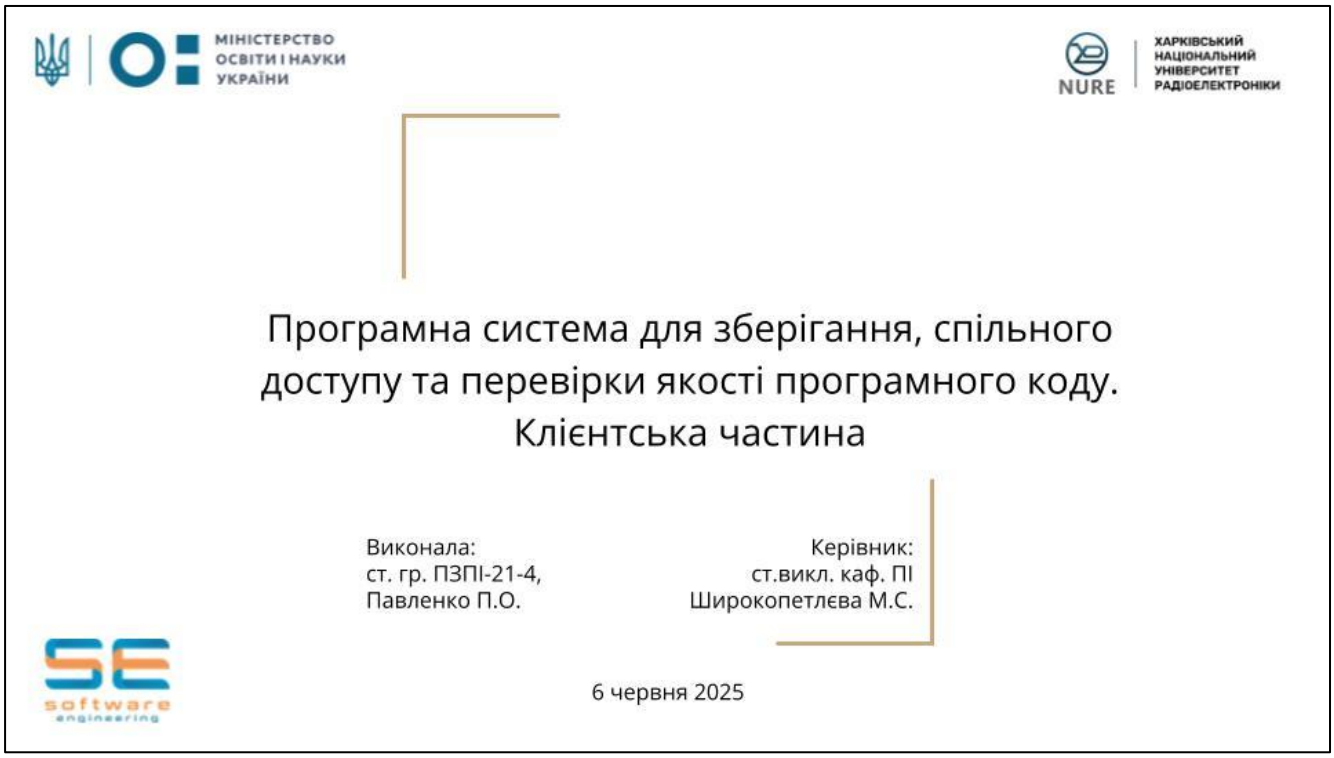
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://web-creator.ru/articles/mvc	11 (1) 0.13 %
2	https://essuir.sumdu.edu.ua/bitstream/123456789/94737/1/Shutylieva_mag_rob.pdf	10 (1) 0.12 %

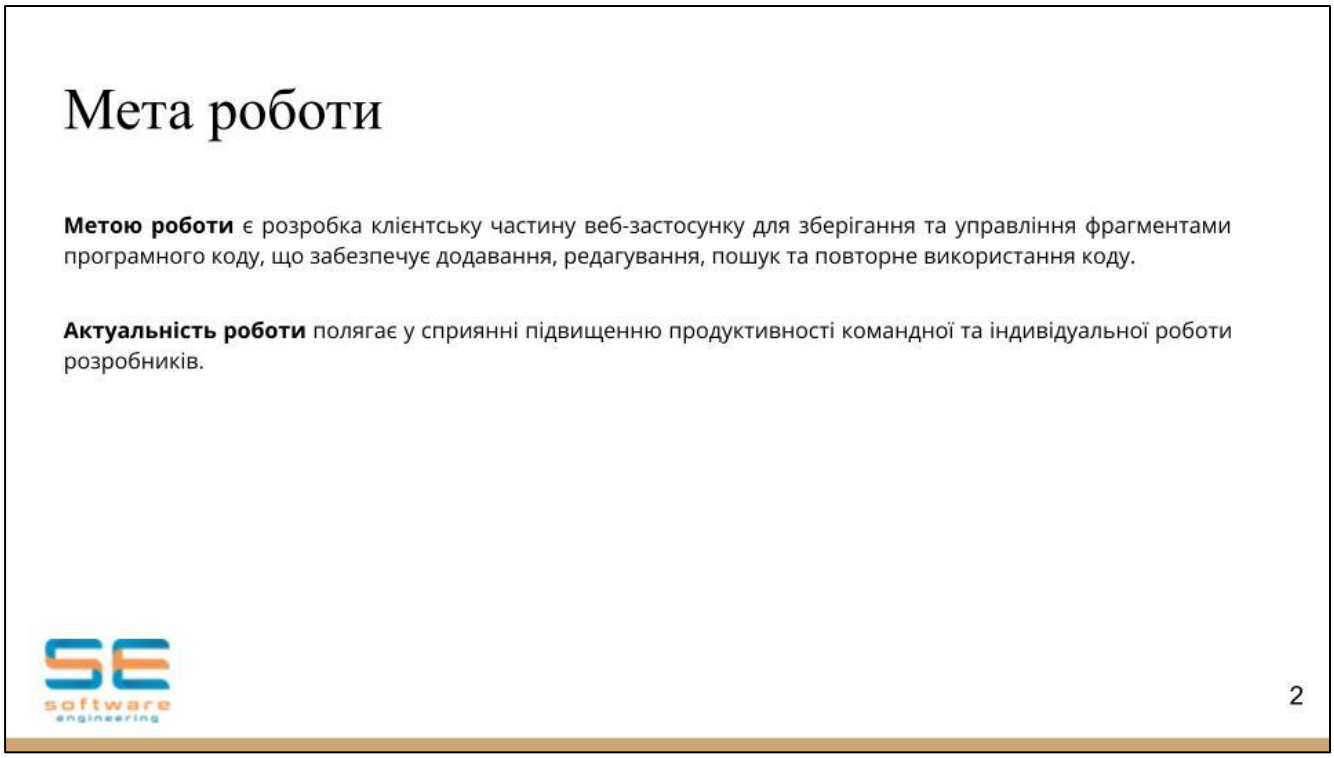
Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

ДОДАТОК Ж

ХНУРЕ Слайди презентації





Аналіз проблеми (аналіз існуючих рішень)

Наведені рішення мають власні переваги, однак не здатні комплексно вирішити проблеми, пов'язані з якісною організацією роботи з фрагментами коду, їх генерацією, перевіркою, зберіганням та пошуком у межах командних чи відкритих середовищ.

Аналог	Використання ШІ для генерації програмних проєктів	Корпоративний доступ до фрагментів коду	Відкритий доступ до коду	Перевірка якості коду	Рівень складності UI / UX
SnippetsLab	–	+	+	+	високий
Code Snippets AI	–	+	–	+	помірний
GitHub Copilot	+	+	–	+	помірний

Постановка задачі та опис системи

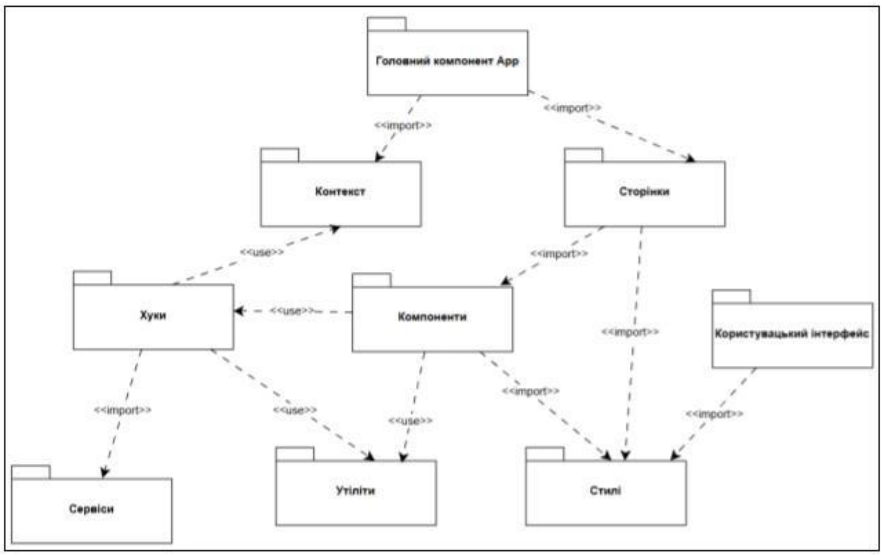
Необхідно **спроєктувати та реалізувати клієнтську частину** системи зберігання та управління фрагментами програмного коду, яка дозволяє ефективно додавати, редагувати, шукати та повторно використовувати код, а також формувати рекомендації щодо його покращення.

Основна проблема, яку має вирішити клієнтська частина системи — це неефективність у процесі збереження та повторного використання фрагментів програмного коду. Клієнтська частина має забезпечити зручні засоби для взаємодії користувача із системою з метою уникнення дублювання коду, оптимізації часу доступу до необхідних фрагментів та покращення загальної якості розробки.

Очікуваний результат - функціональний, зручний інтерфейс, який підвищує продуктивність і якість розробки.

Архітектура клієнтської частини

Клієнтська частина реалізована за **модульною** структурою з використанням **компонентного** підходу. Дані отримуються з API, зберігаються в глобальному стані та оновлюються асинхронно.



5

Опис програмного забезпечення, що було використано у розробці

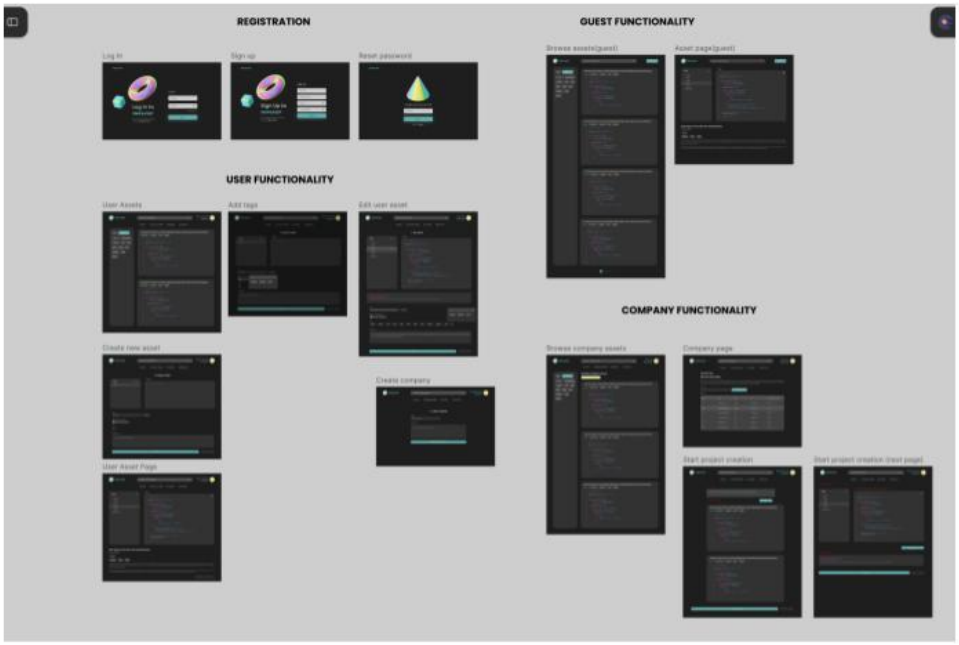
Основні технології та бібліотеки:

- **UML-нотація** — для моделювання структури системи та взаємодії її компонентів;
- **Figma** — для створення макетів та прототипування користувацького інтерфейсу;
- **Visual Studio Code** — як основне середовище розробки з підтримкою розширень для зручної роботи змовою програмування JavaScript;
- **ESLint + Prettier** — для автоматичного форматування та перевірки якості коду;
- **BrowserStack** — для кросбраузерного тестування інтерфейсу на різних пристроях та ОС.
- **JavaScript** – мова програмування клієнтської частини
- **React.js** – бібліотека для побудови користувацького інтерфейсу
- **React Query** – для роботи з API та кешування даних
- **Monaco Editor for React** – інтеграція редактора коду
- **React Responsive** – для реалізації адаптивного дизайну
- Інші **бібліотеки** Radix UI, JWT Decode, React Hook Form, React Hot Toast, React Router DOM

6

Дизайн системи

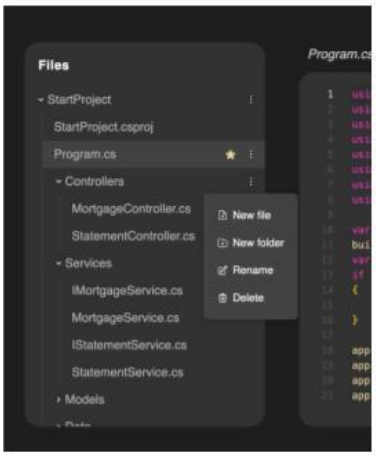
Макет дизайну користувацького інтерфейсу системи, створений у **Figma**



7

Приклад реалізації

Реалізація **файлового дерева**, яке дозволяє створювати, перейменовувати та видаляти файли й папки у вкладеній структурі. Застосовано **рекурсивний** рендеринг компонентів у поєднанні з React-хуками та контекстом для керування станом і взаємодії з користувачем.



8

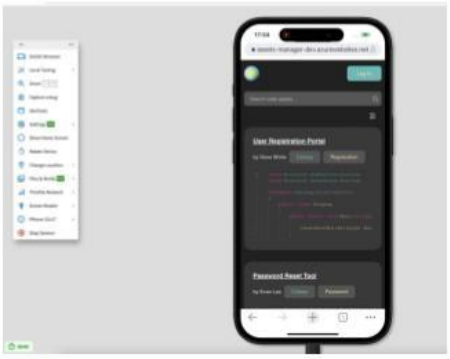
Інтерфейс користувача

9

Тестування

Проведено модульне та респонсивне тестування з використанням **Testing Library**, **Jest-DOM**, **Vitest** та **BrowserStack**.

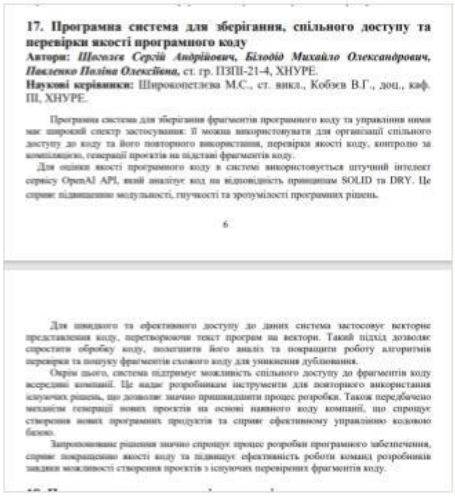
```
✓ src/components/assets/AddAssetForm.test.jsx (6 tests) 145ms
  ✓ AddAssetForm > renders form elements correctly 72ms
  ✓ AddAssetForm > shows error toast if asset name is empty 11ms
  ✓ AddAssetForm > updates name input when user types 10ms
  ✓ AddAssetForm > changes language when selecting from dropdown 21ms
  ✓ AddAssetForm > submits correct asset data for regular user 9ms
  ✓ AddAssetForm > submits correct asset data for enterprise user 20ms

Test Files  1 passed (1)
     Tests  6 passed (6)
  Start at  14:20:28
  Duration  325ms
```



10

Публікація результатів

AI-powered Software Project Generation from Natural Language Input Using Pre-stored Code Assets in Vector Databases



17. Програмна система для зберігання, спільного доступу та перевірки якості програмного коду

Автори: ***Шаглєє Сергій Андрійович, Білодід Михайло Олександрович, Павленко Поліна Олексіївна,*** ст. гр. ПЗПІ-21-4, ХНУРЕ.

Наукові керівники: Широкопетлєва М.С., ст. викл., Кобзєв В.Г., доц., каф. ПІ, ХНУРЕ.

Програмна система для зберігання фрагментів програмного коду та управління ним має широкий спектр застосування: її можна використовувати для організації спільного доступу до коду та його повторного використання, перевірки якості коду, контролю за компіляцією, генерації проєктів на підставі фрагментів коду.

Для оцінки якості програмного коду в системі використовується штучний інтелект сервісу OpenAI API, який аналізує код на відповідність принципам SOLID та DRY. Це сприяє підвищенню модульності, гнучкості та зрозумілості програмних рішень.

6

Для швидкого та ефективного доступу до даних система застосовує векторне представлення коду, перетворюючи текст програм на вектори. Такий підхід дозволяє спрощувати обробку коду, полегшити його аналіз та покращити роботу алгоритмів перевірки та пошуку фрагментів схожого коду для уникнення дублювання.

Окрім цього, система підтримує можливість спільного доступу до фрагментів коду всередині компанії. Це надає розробникам інструменти для повторного використання існуючих рішень, що дозволяє значно пришвидшити процес розробки. Також передбачено механізм генерації нових проєктів на основі наявного коду компанії, що спрощує створення нових програмних продуктів та сприяє ефективному управлінню кодовою базою.

Запропоноване рішення значно спрощує процес розробки програмного забезпечення, сприяє покращенню якості коду та підвищує ефективність роботи команд розробників завдяки можливості створення проєктів з існуючих перевірених фрагментів коду.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ДИПЛОМ

НАГОРОДЖУЄТЬСЯ
Сергій ШОГОЛЄВ, Михайло БІЛОДІД, Поліна ПАВЛЕНКО

за 1 місце

в номінації «Програмне забезпечення. Наука»
на виставці технічної творчості молоді
29-го Міжнародного молодіжного форуму
«Радіоелектроніка та молодь у XXI столітті»

В.о. ректора Юрій РОМАНЕНКОВ
19 квітня 2025 р.

11

Підсумки

- Проведено дослідження проблем управління кодовими базами
- Здійснено аналіз існуючих рішень та визначено ключові труднощі
- Розроблено концептуальну модель програмної системи
- Спроєктовано клієнтську частину системи з урахуванням потреб користувачів
- Реалізовано гнучкий та адаптивний інтерфейс для ефективної роботи з кодовими фрагментами
- Реалізовано клієнтську частину системи для зберігання та управління фрагментами коду
- Виконано тестування функціональності, респонсивності та негативних сценаріїв
- Проведено апробацію результатів через участь у конференціях та виставках
- Є потенціал для подальшого розвитку та масштабування проєкту

12