​​

|  |
| --- |
| ​IITU |
| ​​Report​ |
| ​​Project Title​  ExcelRecover |

​​Name​  Alisher and Jasulan

​​Group​  2406Cs

​

1. **Introduction**

This project is a Python-based utility designed to recover access to protected Excel (.xlsx) files

by intelligently brute-forcing the password and creating an identical, unprotected copy of the original file.

It preserves all contents and structure, providing the user with a usable version of the document when the password is forgotten or lost.

1. **Project Title: ExcelRecover**

**Description:** ExcelRecover is a **Python (PyQt)** application designed to **repair corrupted or recover lost data** from **Microsoft Excel files (.xlsx, .xls)**.

**Key Elements:**

1. **File Loading & Validation**

Supports .xlsx and .xls formats

Detects file corruption and provides error diagnostics

1. **Data Recovery**

Recovers:

Raw cell data (text, numbers, dates)

Formulas (with optional recalculation)

Basic chart objects (if recoverable)

Handles partially readable files with graceful degradation

1. **Output Options**

Save as new Excel file (.xlsx by default)

1. **Error Handling**

Detailed logging for debugging

User-friendly error messages

1. **Objective:**

The ExcelRecover project aims to develop **an open-source desktop application** that provides reliable data recovery for corrupted Microsoft Excel files (\*.xlsx, .xls). Key objectives include:

#### **1. Primary Goal**

To implement a **PyQt-based GUI tool** that:

* Recovers lost/corrupted data while preserving structural integrity
* Surpasses built-in Excel repair capabilities for common corruption scenarios
* Offers better usability than command-line alternatives

#### **2. Academic Objectives**

Demonstrate mastery of:

PyQt5 for cross-platform GUI development

Exception handling in file parsing operations

OO design patterns (Factory pattern for Excel version handling)

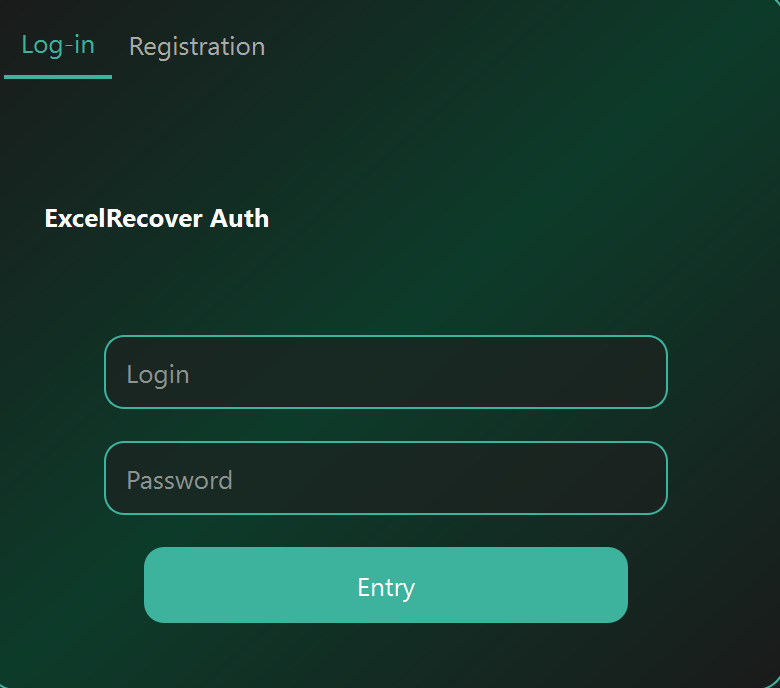
Provide extensible architecture for future enhancements

**Educational and Practical Value:**

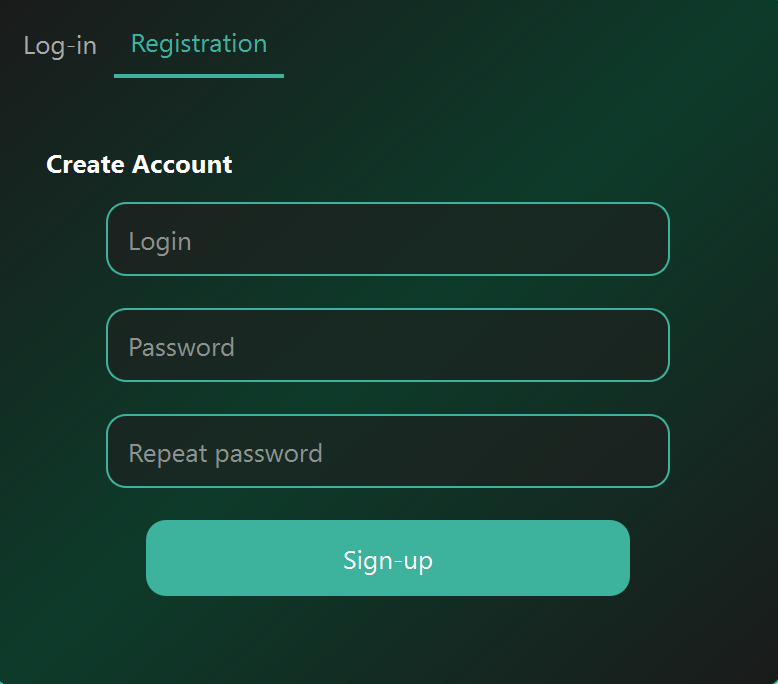
It provides hands-on experience with Python GUI and basic development.

1. **Application window (ScreenShot and description)**

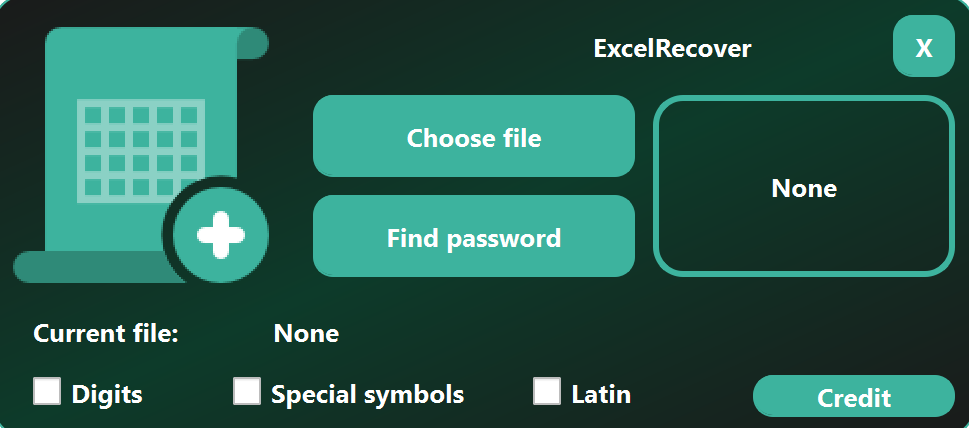
**Auth Windows (1-st window)**



**Auth Windows (2-nd window)**



**Main Menu**



### ****5. Features of the Project****

#### **User-Friendly Interface**

* **Modern Auth System**: Clean login/registration interface with:
  + Input validation (empty fields, password matching)
  + Visual feedback (error messages, success notifications)
  + Responsive design (adapts to window resizing)
* **Intuitive Navigation**:
  + Tab-based switching between login/registration
  + Centered elements and consistent styling (QSS)
  + Animated transitions (optional via QPropertyAnimation)

#### **Task Management (Assuming ExcelRecover is a password recovery tool)**

* **File Processing**:
  + Secure handling of encrypted Excel files (msoffcrypto library)
  + Background thread execution (QThread for non-blocking UI)
* **Password Recovery**:
  + Configurable brute-force settings (digits, symbols, letters)
  + Real-time progress display in UI

#### **Data Persistence**

* **PostgreSQL Integration**:
  + User credentials stored securely (SHA-256 hashing)
  + Schema: users (id, username, password\_hash, created\_at)
  + Activity logging (logs table for audit trail)
* **Session Preservation**:
  + Decrypted files saved as decrypted.xlsx

#### **Error Handling**

* **User-Facing Messages**:
  + Form validation (e.g., "Password must be 6+ characters")
  + Database/File errors (e.g., "File not found", "DB connection failed")
* **Technical Safeguards**:
  + Try-catch blocks for file operations and DB queries
  + Thread-safe signal/slot communication (pyqtSignal)
  + Automated logging (data/log.txt for debugging)

**6. Technology Stack**

* **Programming Language**: Python
* **Framework**:PyQt5
* **IDE**: PyCharm
* **Build Tool**:**Python Interpreter** (python main.py)

**7. Implementation Steps**

1. **Setting Up the Project**

pip install PyQt5 psycopg2-binary python-dotenv msoffcrypto

**PostgreSQL Setup**

Initialized database:

sql

CREATE DATABASE excelrecover\_db;

CREATE USER exceluser WITH PASSWORD '\*\*\*';

GRANT ALL PRIVILEGES ON DATABASE excelrecover\_db TO exceluser;

**Project Structure**

Project ExcelRecover/

├── main.py

├── database.py

├── handler.py

├── des.py

├── .env

└── data/

├── log.txt

└── excel.png

#### **2. Designing the GUI**

**Auth Window (PyQt5)**

**Tools Used**: Qt Designer for .ui file + manual refinements

**Key Components**:

Centered QTabWidget (Login/Register tabs)

Styled QLineEdit fields with validators:

**self.login\_input.setPlaceholderText("Login")**

**self.password\_input.setEchoMode(QtWidgets.QLineEdit.Password)**

Animated QPushButton

css

QPushButton:hover { background: #4ac7b0; }

**Main Window**

File selection dialog:

file = QtWidgets.QFileDialog.getOpenFileName(self, " Choose file ")

**3. Writing the Logic**

#### **3.1 Authentication System**

**Secure User Management**

* **Registration Flow**:
  1. Input validation (check for empty fields, password strength)
  2. Password hashing with SHA-256:

password\_hash = hashlib.sha256(password.encode()).hexdigest()

* 1. Database insertion with duplicate check:

SELECT id FROM users WHERE username = %s", (username,)

if cursor.fetchone():

return False

**Login**:

* 1. Credential verification against PostgreSQL:

"SELECT password\_hash FROM users WHERE username = %s", (username,)

#### **3.2 Excel Password Recovery**

**Brute-Force Engine** (handler.py)

* **Character Set Configuration**:

template = {

0: digits, *# '0123456789'*

1: punctuation, *# '!@#$%^&\*'*

2: ascii\_letters *# 'abc...XYZ'*

}

**Decryption Process**:

1. Attempt decryption with msoffcrypto:

file = msoffcrypto.OfficeFile(open(filepath, "rb"))

file.load\_key(password=attempt)

file.decrypt(open("decrypted.xlsx", "wb"))

1. Success/failure signals:

self.signal.emit(["success", password])

self.signal.emit(["fail", str(e)])

#### **3.3 Data Flow & Error Handling**

**Activity Logging**:

* PostgreSQL logs table tracks:

INSERT INTO logs (user\_id, action, status)

VALUES (1, 'file\_processed', 'success');

**Error Management**:

**User-Facing**:

QtWidgets.QMessageBox.critical(self, "Error", "Invalid file format")

1. **Testing and Debugging**

**1. Verification of operation**

**What was tested:**

**Login and registration**

Verified that a new user can register.

Verified that you can't create two identical logins.

Verified that the correct password is allowed and the wrong password is not.

**Password recovery**

We tested password recovery for a test Excel file.

Checked that the program finds the correct password.

We made sure that the interface doesn't hang when working for a long time.

**Working with the database**

Checked that logins and passwords are saved.

Verified that user actions are logged.

**2. How bugs were found and fixed**

**Common problems:**

**Database connection errors**

Solution: Checked the settings in .env and restarted PostgreSQL.

**Program hangs during password search**

Solution: We added a separate thread (QThread) to prevent the interface from slowing down.

**File is not decrypted**

Solution: Checked that the file is not corrupted and the password is correct.

**How we caught the errors:**

**Wrote errors to a log.txt file to figure out later.**

**Added pop-up messages if something went wrong.**

**3. How we checked that everything worked**

We launched the program and tested all the functions manually.

Checked that you could log in after registering.

Make sure that the password you chose opens the file.

1. **Enhancing the Application (Optional)**

Add the ability to customize the menu style ( switch between dark and light)

1. Code : (main.py)

import os  
import subprocess  
import sys  
import hashlib  
from PyQt5 import QtCore, QtGui, QtWidgets  
from handler import ThreadHandler  
from des import \*  
from database import Database  
  
# Настройки масштабирования  
os.environ["QT\_ENABLE\_HIGHDPI\_SCALING"] = "1"  
os.environ["QT\_SCALE\_FACTOR"] = "1"  
os.environ["QT\_AUTO\_SCREEN\_SCALE\_FACTOR"] = "1"  
  
  
class AuthWindow(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, db):  
 super().\_\_init\_\_()  
 self.db = db  
 self.setup\_ui()  
  
 def setup\_ui(self):  
 self.setWindowTitle("Authorization")  
 self.setFixedSize(400, 350)  
 self.setWindowFlag(QtCore.Qt.FramelessWindowHint)  
 self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)  
 self.center()  
  
 # Стили  
 self.central\_widget = QtWidgets.QWidget(self)  
 self.central\_widget.setObjectName("AuthWidget")  
 self.setCentralWidget(self.central\_widget)  
 self.setStyleSheet("""  
 #AuthWidget {  
 background: qlineargradient(x1:0, y1:0, x2:1, y2:1,  
 stop:0 #1a1a1a, stop:0.5 #0d3b2a, stop:1 #1a1a1a);  
 border-radius: 15px;  
 border: 1px solid #3db39e;  
 }  
  
 QLabel {  
 color: white;  
 font-weight: bold;  
 border: none;  
 background: transparent;  
 text-align: center;  
 }  
  
 QLineEdit {  
 background: rgba(30, 30, 30, 160);  
 color: white;  
 border-radius: 10px;  
 padding: 8px;  
 border: 1px solid #3db39e;  
 margin: 5px 30px;  
 text-align: center;  
 }  
  
 QPushButton {  
 background: #3db39e;  
 color: white;  
 border-radius: 10px;  
 padding: 10px;  
 min-width: 100px;  
 border: none;  
 margin: 5px 50px;  
 }  
 QPushButton:hover {  
 background: #4ac7b0;  
 }  
  
 QTabWidget::pane {  
 border: none;  
 background: transparent;  
 }  
 QTabBar::tab {  
 background: transparent;  
 color: #aaaaaa;  
 padding: 8px;  
 border: none;  
 }  
 QTabBar::tab:selected {  
 color: #3db39e;  
 border-bottom: 2px solid #3db39e;  
 }  
 """)  
  
 # Создаем вкладки  
 self.tabs = QtWidgets.QTabWidget()  
 self.tab\_login = QtWidgets.QWidget()  
 self.tab\_register = QtWidgets.QWidget()  
  
 self.tabs.addTab(self.tab\_login, "Log-in")  
 self.tabs.addTab(self.tab\_register, "Registration")  
  
 # Основной layout  
 layout = QtWidgets.QVBoxLayout(self.central\_widget)  
 layout.addWidget(self.tabs)  
  
 # Настраиваем вкладки  
 self.setup\_login\_tab()  
 self.setup\_register\_tab()  
  
 close\_btn = QtWidgets.QPushButton("✕")  
 close\_btn.setFixedSize(30, 30)  
 close\_btn.setStyleSheet("""  
 background: transparent;  
 border: none;  
 color: #3db39e;  
 font-size: 18px;  
 """)  
 close\_btn.move(self.width() - 40, 10)  
 close\_btn.clicked.connect(self.close)  
  
 def setup\_login\_tab(self):  
 layout = QtWidgets.QVBoxLayout(self.tab\_login)  
 layout.setContentsMargins(20, 20, 20, 20)  
  
 self.login\_input = QtWidgets.QLineEdit()  
 self.login\_input.setPlaceholderText("Login")  
  
 self.password\_input = QtWidgets.QLineEdit()  
 self.password\_input.setPlaceholderText("Password")  
 self.password\_input.setEchoMode(QtWidgets.QLineEdit.Password)  
  
 login\_btn = QtWidgets.QPushButton("Entry")  
 login\_btn.clicked.connect(self.handle\_login)  
  
 layout.addWidget(QtWidgets.QLabel("ExcelRecover Auth"))  
 layout.addWidget(self.login\_input)  
 layout.addWidget(self.password\_input)  
 layout.addWidget(login\_btn)  
  
 def setup\_register\_tab(self):  
 layout = QtWidgets.QVBoxLayout(self.tab\_register)  
 layout.setContentsMargins(20, 20, 20, 20)  
 layout.setAlignment(QtCore.Qt.AlignCenter)  
  
 self.reg\_username = QtWidgets.QLineEdit()  
 self.reg\_username.setPlaceholderText("Login")  
  
 self.reg\_password = QtWidgets.QLineEdit()  
 self.reg\_password.setPlaceholderText("Password")  
 self.reg\_password.setEchoMode(QtWidgets.QLineEdit.Password)  
  
 self.reg\_confirm = QtWidgets.QLineEdit()  
 self.reg\_confirm.setPlaceholderText("Repeat password")  
 self.reg\_confirm.setEchoMode(QtWidgets.QLineEdit.Password)  
  
 register\_btn = QtWidgets.QPushButton("Sign-up")  
 register\_btn.clicked.connect(self.handle\_register)  
  
 layout.addWidget(QtWidgets.QLabel("Create Account"))  
 layout.addWidget(self.reg\_username)  
 layout.addWidget(self.reg\_password)  
 layout.addWidget(self.reg\_confirm)  
 layout.addWidget(register\_btn)  
  
 # --- ОБРАБОТЧИКИ ---  
 def handle\_login(self):  
 username = self.login\_input.text().strip()  
 password = self.password\_input.text().strip()  
  
 if not username or not password:  
 self.show\_error("Fill in all the fields!")  
 return  
  
 if self.db.authenticate\_user(username, password):  
 self.main\_window = MainWindow(self.db)  
 self.main\_window.show()  
 self.close()  
 else:  
 self.show\_error("Invalid login or password")  
  
 def handle\_register(self):  
 username = self.reg\_username.text().strip()  
 password = self.reg\_password.text().strip()  
 confirm = self.reg\_confirm.text().strip()  
  
 if not username or not password or not confirm:  
 self.show\_error("Complete all fields!")  
 return  
  
 if password != confirm:  
 self.show\_error("Passwords don't match!")  
 return  
  
 if len(password) < 6:  
 self.show\_error("Password must be 6 characters or more")  
 return  
  
 if self.db.create\_user(username, password):  
 self.show\_success("Registration successful! Now log in")  
 self.tabs.setCurrentIndex(0) # Переключаем на вкладку входа  
 self.login\_input.setText(username) # Подставляем логин  
 self.password\_input.clear()  
 else:  
 self.show\_error("Login's already taken")  
  
 def show\_error(self, message):  
 error\_box = QtWidgets.QMessageBox()  
 error\_box.setIcon(QtWidgets.QMessageBox.Warning)  
 error\_box.setText(message)  
 error\_box.setWindowTitle("Error")  
 error\_box.exec\_()  
  
 def show\_success(self, message):  
 msg = QtWidgets.QMessageBox()  
 msg.setIcon(QtWidgets.QMessageBox.Information)  
 msg.setText(message)  
 msg.setWindowTitle("Success")  
 msg.exec\_()  
  
 def center(self):  
 qr = self.frameGeometry()  
 cp = QtWidgets.QDesktopWidget().availableGeometry().center()  
 qr.moveCenter(cp)  
 self.move(qr.topLeft())  
  
 def mousePressEvent(self, event):  
 self.oldPos = event.globalPos()  
  
 def mouseMoveEvent(self, event):  
 delta = QtCore.QPoint(event.globalPos() - self.oldPos)  
 self.move(self.x() + delta.x(), self.y() + delta.y())  
 self.oldPos = event.globalPos()  
  
class MainWindow(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, db):  
 super().\_\_init\_\_()  
 self.db = db  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 # размер главного меню  
 self.setFixedSize(510, 240)  
 self.setWindowFlag(QtCore.Qt.FramelessWindowHint)  
 self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)  
 self.center()  
  
 # Css для гоавного меню  
 self.ui.frame.setStyleSheet("""  
 QFrame {  
 background: qlineargradient(x1:0, y1:0, x2:1, y2:1,  
 stop:0 #1a1a1a, stop:0.5 #0d3b2a, stop:1 #1a1a1a);  
 border-radius: 15px;  
 border: 1px solid #3db39e;  
 }  
 QLabel {  
 color: white;  
 background: transparent;  
 border: none;  
 }  
 QCheckBox {  
 color: white;  
 }  
 QPushButton {  
 background: qlineargradient(x1:0, y1:0, x2:1, y2:0,  
 stop:0 #2d3534, stop:0.5 #3db39e, stop:1 #2d3534);  
 color: white;  
 border-radius: 10px;  
 border: 1px solid #3db39e;  
 }  
 QPushButton:hover {  
 background: qlineargradient(x1:0, y1:0, x2:1, y2:0,  
 stop:0 #3a4a45, stop:0.5 #4ac7b0, stop:1 #3a4a45);  
 }  
 """)  
  
 # Инициализация данных  
 self.excel\_file = None  
  
  
  
 self.ui.pushButton\_4.clicked.connect(self.show\_credits)  
 self.ui.pushButton\_3.clicked.connect(self.close)  
 self.ui.pushButton.clicked.connect(self.choose\_file)  
 self.ui.pushButton\_2.clicked.connect(self.start\_process)  
  
 # Обработчик потока  
 self.handler = ThreadHandler()  
 self.handler.signal.connect(self.signal\_handler)  
  
 def center(self):  
 qr = self.frameGeometry()  
 cp = QtWidgets.QDesktopWidget().availableGeometry().center()  
 qr.moveCenter(cp)  
 self.move(qr.topLeft())  
  
 def mousePressEvent(self, event):  
 self.oldPos = event.globalPos()  
  
 def mouseMoveEvent(self, event):  
 delta = QtCore.QPoint(event.globalPos() - self.oldPos)  
 self.move(self.x() + delta.x(), self.y() + delta.y())  
 self.oldPos = event.globalPos()  
  
 def choose\_file(self):  
 file = QtWidgets.QFileDialog.getOpenFileName(self)[0]  
 if file:  
 self.excel\_file = file  
 self.ui.label\_4.setText(os.path.basename(self.excel\_file))  
  
 def start\_process(self):  
 conf\_list = [  
 self.ui.checkBox.isChecked(),  
 self.ui.checkBox\_2.isChecked(),  
 self.ui.checkBox\_3.isChecked(),  
 ]  
  
 if self.excel\_file:  
 self.handler.filepath = self.excel\_file  
 self.handler.config = conf\_list  
  
 if any(conf\_list):  
 self.handler.start()  
 self.ui.pushButton.setDisabled(True)  
 self.ui.pushButton\_2.setDisabled(True)  
 else:  
 QtWidgets.QMessageBox.warning(self, "Error", "Configuration must be set up")  
 else:  
 QtWidgets.QMessageBox.warning(self, "Error", "You must select a file")  
  
 def signal\_handler(self, value):  
 if value[0] == "result":  
 self.ui.label\_3.setText(value[1])  
 QtWidgets.QMessageBox.about(self, "Success",  
 f"Password recovered: {value[1]}\nFile created: decrypted.xlsx")  
 self.ui.pushButton.setDisabled(False)  
 self.ui.pushButton\_2.setDisabled(False)  
 elif value[0] == "fail":  
 self.ui.label\_3.setText(value[1])  
  
 def show\_credits(self):  
 credit\_path = r"C:\Users\Alisher\Desktop\Project ExcelRecover\credit.txt"  
 try:  
 if os.path.exists(credit\_path):  
 if sys.platform == "win32":  
 os.startfile(credit\_path)  
 else:  
 opener = "open" if sys.platform == "darwin" else "xdg-open"  
 subprocess.call([opener, credit\_path])  
 else:  
 QtWidgets.QMessageBox.warning(self, "Error", "credit.txt is not found. ")  
 except Exception as e:  
 QtWidgets.QMessageBox.warning(self, "Error", f"Failed to open file: {str(e)}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QtWidgets.QApplication(sys.argv)  
  
 # Настройки HighDPI  
 if hasattr(QtCore.Qt, 'AA\_EnableHighDpiScaling'):  
 QtWidgets.QApplication.setAttribute(QtCore.Qt.AA\_EnableHighDpiScaling, True)  
 if hasattr(QtCore.Qt, 'AA\_UseHighDpiPixmaps'):  
 QtWidgets.QApplication.setAttribute(QtCore.Qt.AA\_UseHighDpiPixmaps, True)  
  
 app.setStyle('Fusion')  
 db = Database()  
 if not db.conn:  
 print("PostgreSQL 17 connection error!")  
 sys.exit(1)  
  
 # Шрифт  
 font = QtGui.QFont()  
 font.setFamily("Segoe UI")  
 font.setPointSize(10)  
 app.setFont(font)  
  
 # Создаем и показываем окно авторизации  
 auth\_window = AuthWindow(db)  
 auth\_window.show()  
  
 sys.exit(app.exec\_())

untitled.ui

<?xml version="1.0" encoding="UTF-8"?>  
<ui version="4.0">  
 <class>MainWindow</class>  
 <widget class="QMainWindow" name="MainWindow">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>510</width>  
 <height>240</height>  
 </rect>  
 </property>  
 <property name="windowTitle">  
 <string>MainWindow</string>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <widget class="QFrame" name="frame">  
 <property name="geometry">  
 <rect>  
 <x>10</x>  
 <y>10</y>  
 <width>491</width>  
 <height>221</height>  
 </rect>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QFrame {  
 border-radius: 15px;  
 background-color: #394341;  
}</string>  
 </property>  
 <property name="frameShape">  
 <enum>QFrame::StyledPanel</enum>  
 </property>  
 <property name="frameShadow">  
 <enum>QFrame::Raised</enum>  
 </property>  
 <widget class="QLabel" name="label\_2">  
 <property name="geometry">  
 <rect>  
 <x>10</x>  
 <y>10</y>  
 <width>131</width>  
 <height>141</height>  
 </rect>  
 </property>  
 <property name="text">  
 <string/>  
 </property>  
 <property name="pixmap">  
 <pixmap>data/excel.png</pixmap>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="pushButton">  
 <property name="geometry">  
 <rect>  
 <x>160</x>  
 <y>50</y>  
 <width>161</width>  
 <height>41</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QPushButton {  
 border-radius: 10px;  
 background-color: #3db39e;  
 color: white;  
}  
  
QPushButton:hover {  
 border-radius: 10px;  
 background-color: #3ca492;  
 color: white;  
}  
  
QPushButton:pressed {  
 border-radius: 10px;  
 background-color: #3d9888;  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Выбрать файл</string>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="pushButton\_2">  
 <property name="geometry">  
 <rect>  
 <x>160</x>  
 <y>100</y>  
 <width>161</width>  
 <height>41</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QPushButton {  
 border-radius: 10px;  
 background-color: #3db39e;  
 color: white;  
}  
  
QPushButton:hover {  
 border-radius: 10px;  
 background-color: #3ca492;  
 color: white;  
}  
  
QPushButton:pressed {  
 border-radius: 10px;  
 background-color: #3d9888;  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Подобрать пароль</string>  
 </property>  
 </widget>  
 <widget class="QLabel" name="label">  
 <property name="geometry">  
 <rect>  
 <x>20</x>  
 <y>160</y>  
 <width>111</width>  
 <height>16</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QLabel {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Текущий файл:</string>  
 </property>  
 </widget>  
 <widget class="QLabel" name="label\_4">  
 <property name="geometry">  
 <rect>  
 <x>140</x>  
 <y>160</y>  
 <width>441</width>  
 <height>16</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QLabel {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>None</string>  
 </property>  
 </widget>  
 <widget class="QPushButton" name="pushButton\_3">  
 <property name="geometry">  
 <rect>  
 <x>450</x>  
 <y>10</y>  
 <width>31</width>  
 <height>31</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QPushButton {  
 border-radius: 10px;  
 background-color: #3db39e;  
 color: white;  
}  
  
QPushButton:hover {  
 border-radius: 10px;  
 background-color: #3ca492;  
 color: white;  
}  
  
QPushButton:pressed {  
 border-radius: 10px;  
 background-color: #3d9888;  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>X</string>  
 </property>  
 </widget>  
 <widget class="QLabel" name="label\_5">  
 <property name="geometry">  
 <rect>  
 <x>300</x>  
 <y>15</y>  
 <width>141</width>  
 <height>21</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QLabel {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>ExcelRecover</string>  
 </property>  
 </widget>  
 <widget class="QCheckBox" name="checkBox">  
 <property name="geometry">  
 <rect>  
 <x>20</x>  
 <y>190</y>  
 <width>71</width>  
 <height>17</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QCheckBox {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Цифры</string>  
 </property>  
 </widget>  
 <widget class="QCheckBox" name="checkBox\_2">  
 <property name="geometry">  
 <rect>  
 <x>120</x>  
 <y>190</y>  
 <width>121</width>  
 <height>17</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QCheckBox {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Спец.символы</string>  
 </property>  
 </widget>  
 <widget class="QCheckBox" name="checkBox\_3">  
 <property name="geometry">  
 <rect>  
 <x>270</x>  
 <y>190</y>  
 <width>91</width>  
 <height>17</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="cursor">  
 <cursorShape>PointingHandCursor</cursorShape>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QCheckBox {  
 color: white;  
}</string>  
 </property>  
 <property name="text">  
 <string>Латиница</string>  
 </property>  
 </widget>  
 <widget class="QLabel" name="label\_3">  
 <property name="geometry">  
 <rect>  
 <x>330</x>  
 <y>50</y>  
 <width>151</width>  
 <height>91</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>10</pointsize>  
 <weight>75</weight>  
 <bold>true</bold>  
 </font>  
 </property>  
 <property name="acceptDrops">  
 <bool>false</bool>  
 </property>  
 <property name="styleSheet">  
 <string notr="true">QLabel {  
 color: white;  
 border: 3px solid #3db39e;  
}</string>  
 </property>  
 <property name="frameShape">  
 <enum>QFrame::NoFrame</enum>  
 </property>  
 <property name="frameShadow">  
 <enum>QFrame::Plain</enum>  
 </property>  
 <property name="lineWidth">  
 <number>1</number>  
 </property>  
 <property name="midLineWidth">  
 <number>0</number>  
 </property>  
 <property name="text">  
 <string>None</string>  
 </property>  
 <property name="textFormat">  
 <enum>Qt::AutoText</enum>  
 </property>  
 <property name="scaledContents">  
 <bool>false</bool>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 <property name="wordWrap">  
 <bool>false</bool>  
 </property>  
 <property name="openExternalLinks">  
 <bool>false</bool>  
 </property>  
 </widget>  
 </widget>  
 </widget>  
 </widget>  
 <resources/>  
 <connections/>  
</ui>

database.py

import os  
import psycopg2  
from dotenv import load\_dotenv  
import hashlib  
  
load\_dotenv()  
  
  
class Database:  
 def \_\_init\_\_(self):  
 self.conn = None  
 self.connect()  
  
 def connect(self):  
 try:  
 self.conn = psycopg2.connect(  
 host=os.getenv('DB\_HOST'),  
 port=os.getenv('DB\_PORT'),  
 dbname=os.getenv('DB\_NAME'),  
 user=os.getenv('DB\_USER'),  
 password=os.getenv('DB\_PASSWORD')  
 )  
 self.\_init\_db()  
 print(" Подключение к БД успешно!")  
 except Exception as e:  
 print(f" Ошибка подключения: {e}")  
  
 def \_init\_db(self):  
 with self.conn.cursor() as cur:  
 cur.execute("""  
 CREATE TABLE IF NOT EXISTS users (  
 id SERIAL PRIMARY KEY,  
 username VARCHAR(50) UNIQUE NOT NULL,  
 password\_hash VARCHAR(255) NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
 )  
 """)  
 self.conn.commit()  
  
 def create\_user(self, username, password):  
 try:  
 with self.conn.cursor() as cur:  
 # Проверяем, не занят ли логин  
 cur.execute("SELECT id FROM users WHERE username = %s", (username,))  
 if cur.fetchone():  
 return False # Логин занят  
  
 password\_hash = hashlib.sha256(password.encode()).hexdigest()  
  
 # Сохраняем в БД  
 cur.execute(  
 "INSERT INTO users (username, password\_hash) VALUES (%s, %s)",  
 (username, password\_hash)  
 )  
 self.conn.commit()  
 return True  
 except Exception as e:  
 print(f"Ошибка регистрации: {e}")  
 return False  
  
 def authenticate\_user(self, username, password):  
 #Проверяем логин и пароль  
 try:  
 with self.conn.cursor() as cur:  
 cur.execute(  
 "SELECT password\_hash FROM users WHERE username = %s",  
 (username,)  
 )  
 result = cur.fetchone()  
 if result:  
 # Сравниваем хеши паролей  
 return result[0] == hashlib.sha256(password.encode()).hexdigest()  
 return False  
 except Exception as e:  
 print(f"Ошибка входа: {e}")  
 return False  
  
 def close(self):  
 if self.conn:  
 self.conn.close()

**Challenges Faced:**

#### **1. Database Connection Issues**

**Problem**: The app sometimes failed to connect to PostgreSQL.

**Solution**:

Double-checked .env file credentials.

Added error messages to help users troubleshoot.

Tested with different PostgreSQL versions (15/16/17).

**2. Password recovery speed**

**Problem:** Forced password recovery was taking too long, causing the user interface to freeze.

**Solution:**

Used QThread to perform decryption in the background.

**Conclustion:  
  
  
  
Conclusion**

**ExcelRecover project has successfully solved the task of recovering forgotten passwords from Excel files. Here are the main results:**

**What we got**

**A working tool that helps to recover data from Excel documents**

**Recovers passwords to .xlsx files**

**Saves user data to PostgreSQL database**

**User-friendly interface**

**Simple login and registration windows**

**Does not hang when working for a long time**

**Error protection**

**Checks input data**

**Saves logs for diagnosing problems**

**What difficulties were encountered**

**Connecting to database**

**Solution: Checked settings**

**Incorrect user actions**

**Solution: Added hints and input validation**

**What could be improved**

**Switching between light and dark theme**

**Bottom line: The program is ready to use, works stably and solves its main task.**