МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет физики, математики, информатики

Кафедра программного обеспечения и администрирования информационных систем

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

Интерфейсы программирования приложений

на тему: *Проектирование пользовательского интерфейса десктопного приложения*

Обучающегося 2 курса очной формы обучения

направления подготовки

09.03.01 Информатика и вычислительная техника

Направленность (профиль)

Прикладной искусственный интеллект

Гордеева Ярослава Сергеевича

Руководитель:

старший преподаватель кафедры ПОАИС

Ураева Елена Евгеньевна

Курск, 2025

# **Индивидуальное задание**

1. Разработать пользовательский интерфейс для десктопного клиента, на основе макета, описанного в лабораторной работе. При необходимости актуализировать карту навигации и черновой прототип с учетом вида интерфейса.
2. На основании одной из стилистик разработать дизайн концепцию приложения.
3. Создать проект в среде разработке согласно разработанному интерфейсу. Обеспечить переходы между экранами приложения согласно навигационной карте при выполнении действий пользователя по описанным сценариям. Допустимо использовать в проекте статичные данные для элементов вывода.
4. Разработать систему оповещений для альтернативных веток сценария.

# **Дизайн концепция приложения**

На рисунке 1 представлена главная страница программы.

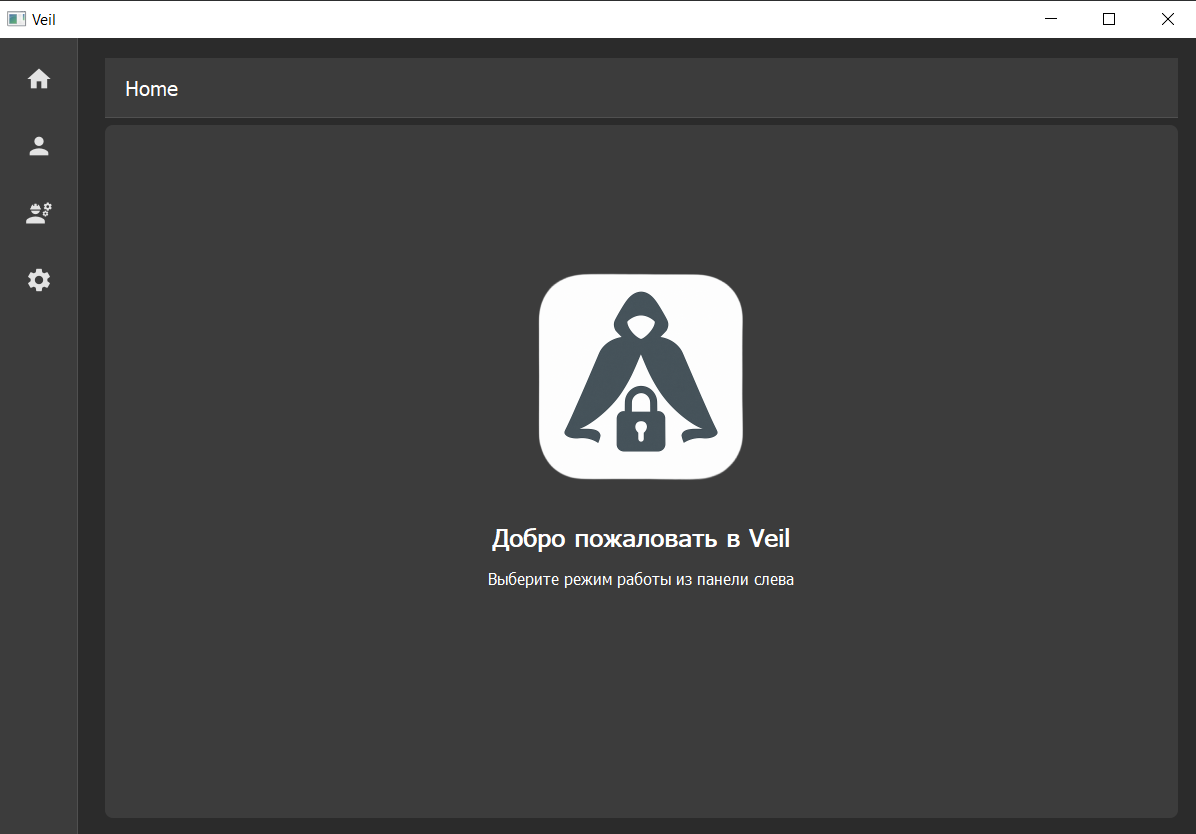


Рисунок 1 – Вид главной страницы программы

Страница для обычного режима отображена на рисунке 2.

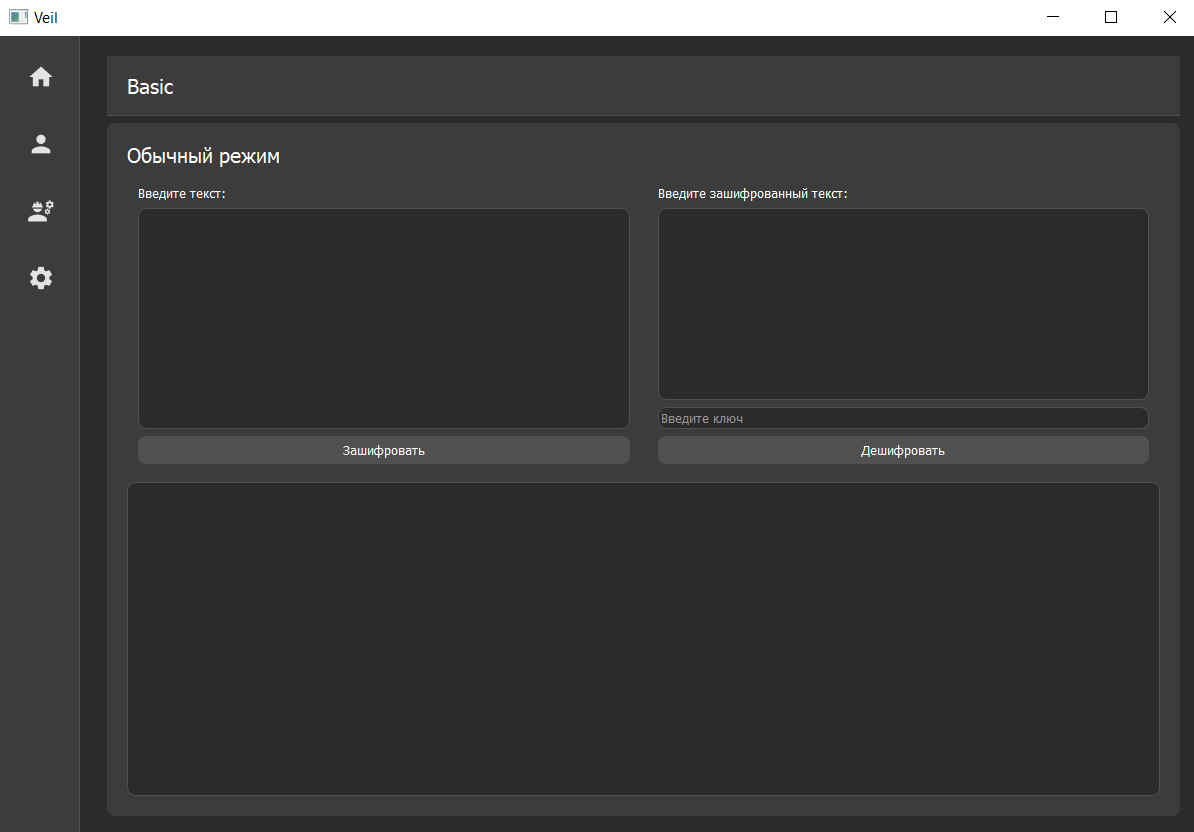


Рисунок 2 – Вид страницы обычного режима

На рисунке 3 отображен функционал обычного режима.

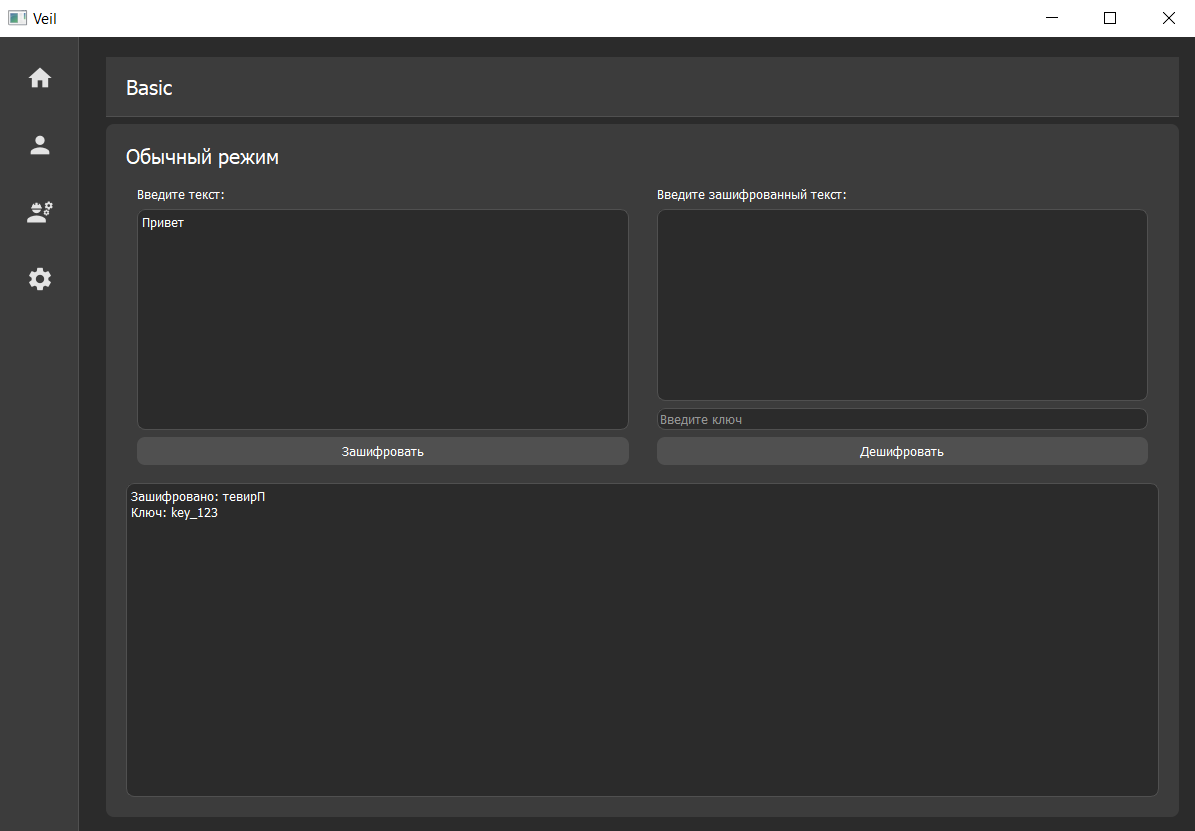


Рисунок 3 – Функционал обычного режима

На рисунке 4 отображена страница продвинутого режима.

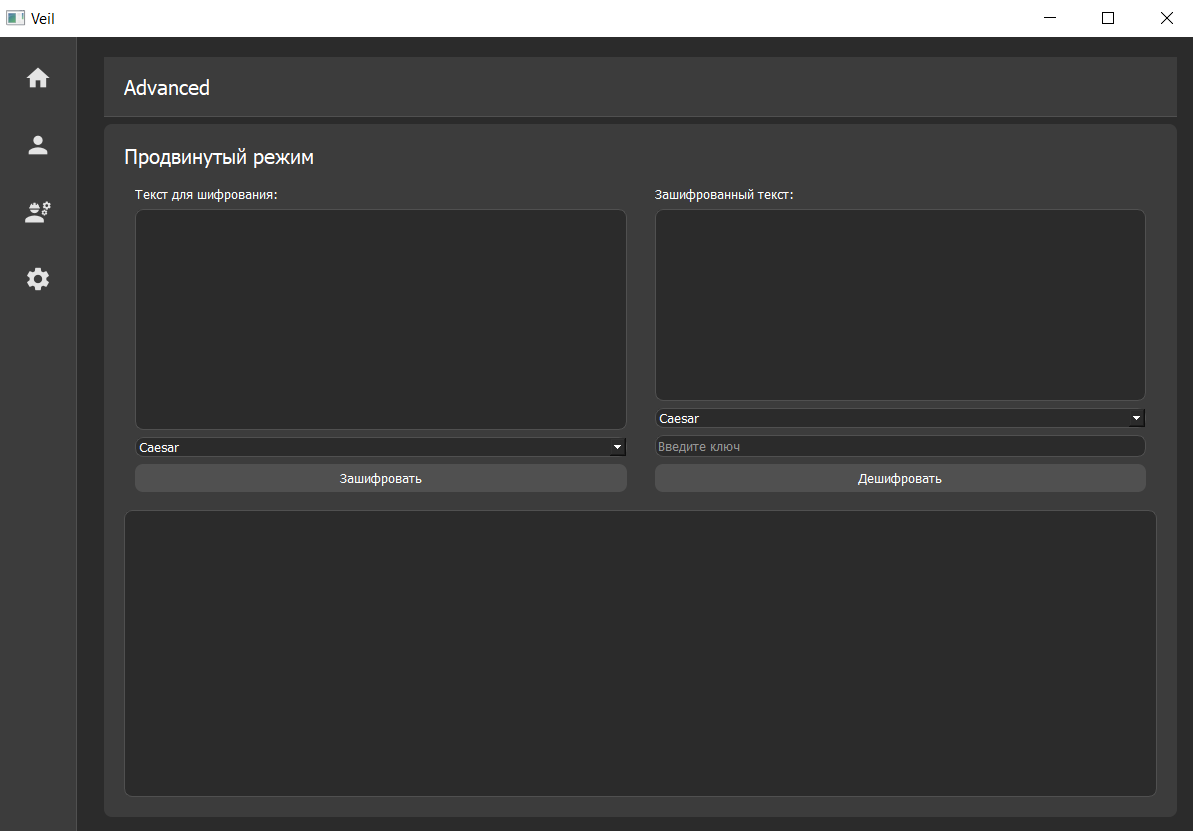


Рисунок 4 – Страница продвинутого режима

На рисунке 5 отображен функционал продвинутого режима.

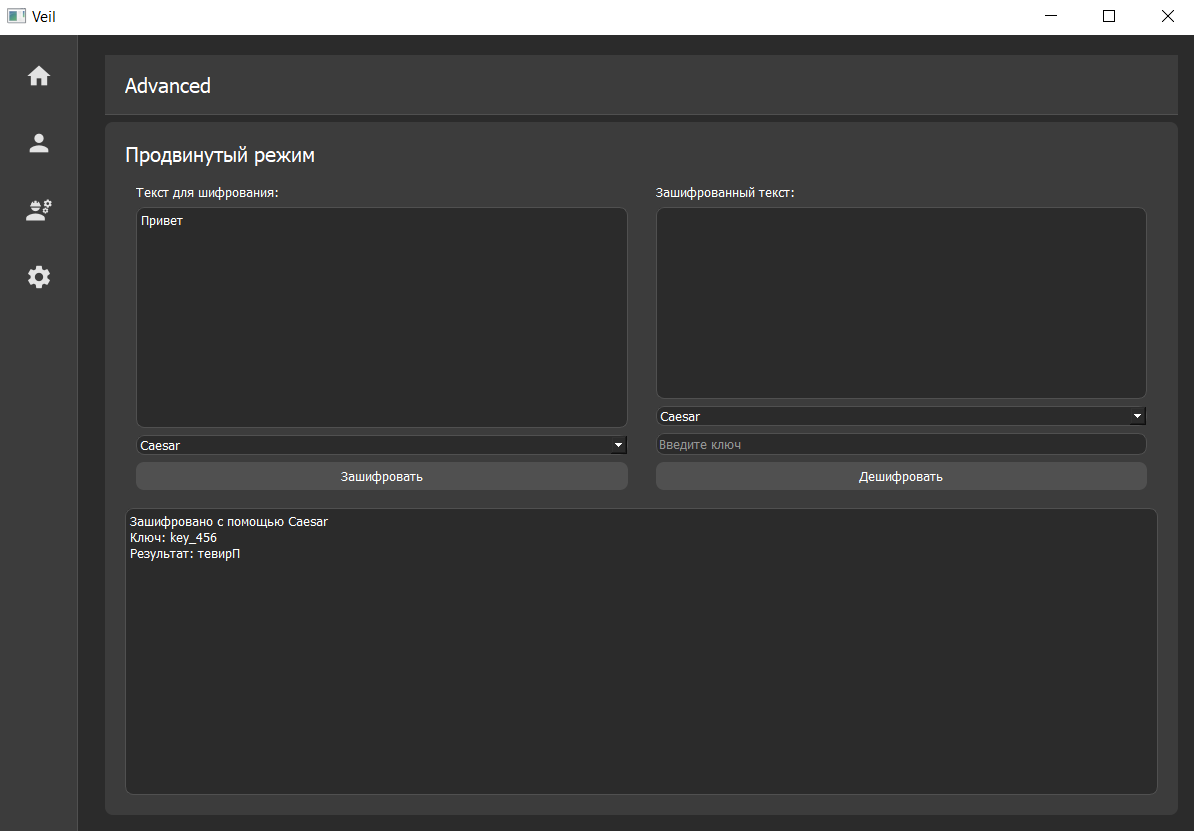


Рисунок 5 – Функционал продвинутого режима

На рисунке 6 отображена страница настроек.

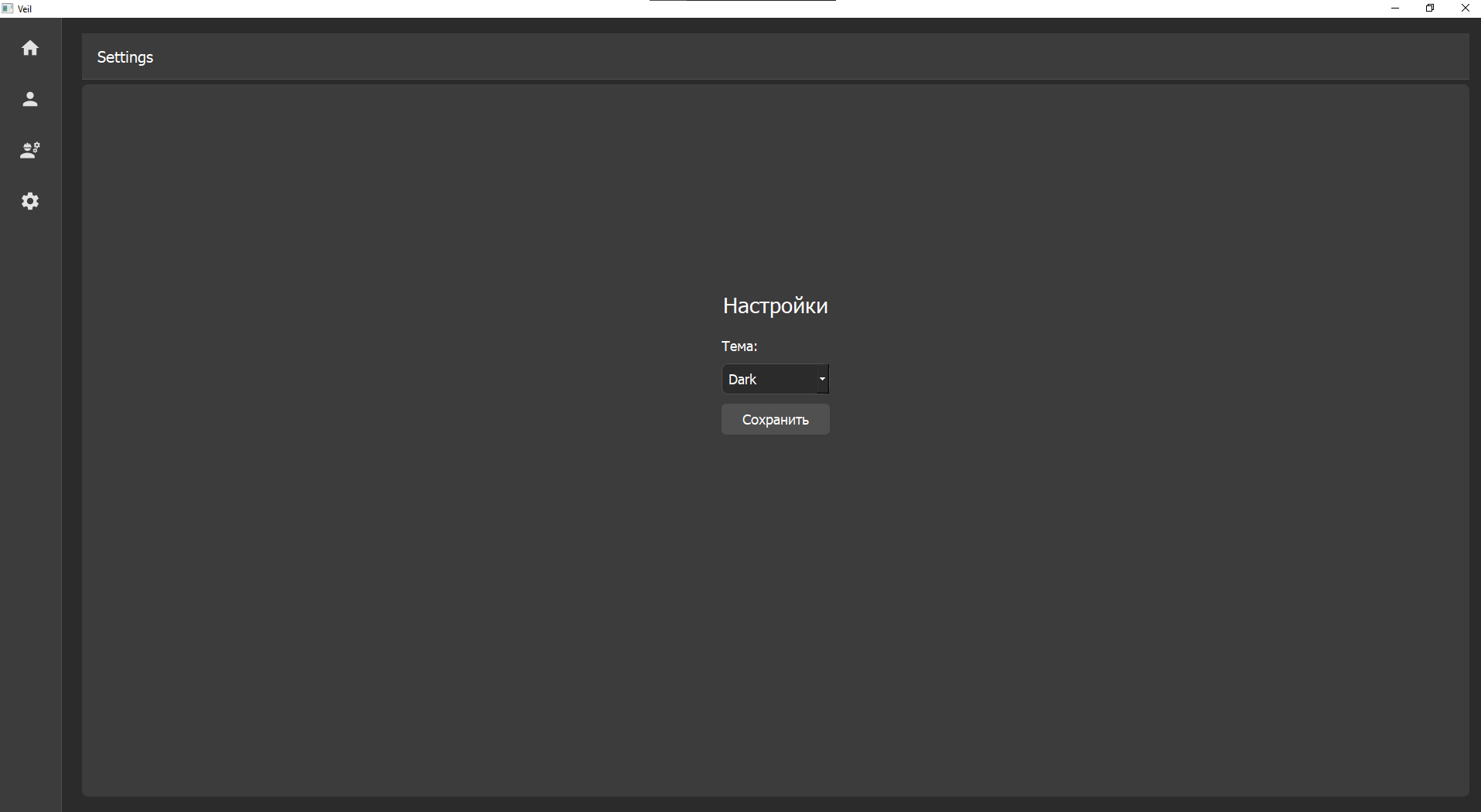


Рисунок 6 – Вид страницы настроек

# **Проект в среде разработки**

Ниже представлен код программы в среде разработки Python:

import sys  
  
from PyQt5.QtWidgets import (  
 QApplication, QMainWindow, QWidget, QFrame, QLabel, QPushButton,  
 QHBoxLayout, QVBoxLayout, QStackedWidget,  
 QFormLayout, QComboBox, QMessageBox,  
 QLineEdit, QTextEdit, QSplitter  
)  
from PyQt5.QtCore import (  
 Qt, QSize, QPoint, QEasingCurve,  
 QPropertyAnimation, QParallelAnimationGroup  
)  
from PyQt5.QtGui import QIcon, QFontDatabase, QFont, QPixmap  
  
  
DARK\_BG = "#2B2B2B"  
CARD\_BG = "#3C3C3C"  
BORDER = "#505050"  
TEXT\_LIGHT = "#FFFFFF"  
  
class AnimatedStackedWidget(QStackedWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 def setCurrentIndex(self, index):  
 self.slideInIndex(index)  
  
 def slideInIndex(self, index):  
 count = self.count()  
 if index < 0:  
 index = 0  
 elif index >= count:  
 index = count - 1  
  
 current = self.currentWidget()  
 next\_widget = self.widget(index)  
 offset = self.width()  
  
 next\_widget.setGeometry(self.rect())  
 next\_widget.move(offset, 0)  
 next\_widget.show()  
 next\_widget.raise\_()  
  
 anim\_group = QParallelAnimationGroup(self)  
 anim1 = QPropertyAnimation(current, b"pos", self)  
 anim1.setDuration(250)  
 anim1.setStartValue(QPoint(0, 0))  
 anim1.setEndValue(QPoint(-offset, 0))  
 anim1.setEasingCurve(QEasingCurve.OutSine)  
 anim\_group.addAnimation(anim1)  
  
 anim2 = QPropertyAnimation(next\_widget, b"pos", self)  
 anim2.setDuration(250)  
 anim2.setStartValue(QPoint(offset, 0))  
 anim2.setEndValue(QPoint(0, 0))  
 anim2.setEasingCurve(QEasingCurve.OutSine)  
 anim\_group.addAnimation(anim2)  
  
 anim\_group.finished.connect(lambda idx=index: QStackedWidget.setCurrentIndex(self, idx))  
 anim\_group.start()  
  
class SideBarButton(QPushButton):  
 def \_\_init\_\_(self, icon\_path, tooltip):  
 super().\_\_init\_\_()  
 self.setFixedSize(60, 60)  
 self.setIcon(QIcon(icon\_path))  
 self.setIconSize(QSize(28, 28))  
 self.setToolTip(tooltip)  
 self.setCursor(Qt.PointingHandCursor)  
 self.setStyleSheet(f"""  
 QPushButton {{  
 background: transparent;  
 border: none;  
 }}  
 QPushButton:hover {{  
 background: {BORDER};  
 }}  
 """)  
  
class TitleBar(QFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setFixedHeight(60)  
 self.setStyleSheet(f"background: {CARD\_BG}; border-bottom: 1px solid {BORDER};")  
 layout = QHBoxLayout(self)  
 layout.setContentsMargins(20, 0, 20, 0)  
 self.title = QLabel("Home")  
 self.title.setStyleSheet(f"font-size: 20px; color: {TEXT\_LIGHT}; font-weight: 300;")  
 layout.addWidget(self.title)  
 layout.addStretch()  
  
class ContentFrame(QFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setStyleSheet(f"background: {CARD\_BG}; border-radius: 8px;")  
 self.layout = QVBoxLayout(self)  
 self.layout.setContentsMargins(20, 20, 20, 20)  
  
 def addWidget(self, widget):  
 self.layout.addWidget(widget)  
  
 def addLayout(self, layout):  
 self.layout.addLayout(layout)  
  
class HomeScreen(ContentFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 image\_container = QLabel()  
 pixmap = QPixmap("icons/logo.png")  
 pixmap = pixmap.scaled(256, 256, Qt.KeepAspectRatio, Qt.SmoothTransformation)  
 image\_container.setPixmap(pixmap)  
 image\_container.setAlignment(Qt.AlignCenter)  
  
 lbl = QLabel("Добро пожаловать в Veil")  
 lbl.setStyleSheet(f"font-size: 24px; color: {TEXT\_LIGHT}; font-weight: 500;")  
 lbl.setAlignment(Qt.AlignCenter)  
  
 instructions = QLabel("Выберите режим работы из панели слева")  
 instructions.setStyleSheet(f"font-size: 16px; color: {TEXT\_LIGHT}; font-weight: 200;")  
 instructions.setAlignment(Qt.AlignCenter)  
  
 self.layout.addStretch(1)  
 self.addWidget(image\_container)  
 self.layout.addSpacing(10)  
 self.addWidget(lbl)  
 self.layout.addSpacing(10)  
 self.addWidget(instructions)  
 self.layout.addStretch(2)  
 self.layout.setAlignment(Qt.AlignCenter)  
  
class BasicScreen(ContentFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 lbl = QLabel("Обычный режим")  
 lbl.setStyleSheet(f"font-size: 20px; color: {TEXT\_LIGHT}; font-weight: 300;")  
 self.addWidget(lbl)  
  
 enc\_frame = QFrame()  
 enc\_layout = QVBoxLayout(enc\_frame)  
 lbl1 = QLabel("Введите текст:")  
 lbl1.setStyleSheet(f"color: {TEXT\_LIGHT};")  
 enc\_layout.addWidget(lbl1)  
 self.basic\_text = QTextEdit()  
 self.basic\_text.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 enc\_layout.addWidget(self.basic\_text)  
 btn\_enc = QPushButton("Зашифровать")  
 btn\_enc.setStyleSheet(f"background: {BORDER}; color: {TEXT\_LIGHT}; border: none; padding: 6px;")  
 btn\_enc.clicked.connect(self.basic\_encrypt)  
 enc\_layout.addWidget(btn\_enc)  
  
 dec\_frame = QFrame()  
 dec\_layout = QVBoxLayout(dec\_frame)  
 lbl2 = QLabel("Введите зашифрованный текст:")  
 lbl2.setStyleSheet(f"color: {TEXT\_LIGHT};")  
 dec\_layout.addWidget(lbl2)  
 self.basic\_decrypt\_text = QTextEdit()  
 self.basic\_decrypt\_text.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 dec\_layout.addWidget(self.basic\_decrypt\_text)  
 self.basic\_key = QLineEdit()  
 self.basic\_key.setPlaceholderText("Введите ключ")  
 self.basic\_key.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 dec\_layout.addWidget(self.basic\_key)  
 btn\_dec = QPushButton("Дешифровать")  
 btn\_dec.setStyleSheet(f"background: {BORDER}; color: {TEXT\_LIGHT}; border: none; padding: 6px;")  
 btn\_dec.clicked.connect(self.basic\_decrypt)  
 dec\_layout.addWidget(btn\_dec)  
  
 splitter = QSplitter()  
 splitter.addWidget(enc\_frame)  
 splitter.addWidget(dec\_frame)  
 self.addWidget(splitter)  
  
 self.result = QTextEdit()  
 self.result.setReadOnly(True)  
 self.result.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 self.addWidget(self.result)  
  
 def basic\_encrypt(self):  
 text = self.basic\_text.toPlainText()  
 encrypted = text[::-1]  
 dummy\_key = "key\_123"  
 self.result.setText(f"Зашифровано: {encrypted}\nКлюч: {dummy\_key}")  
  
 def basic\_decrypt(self):  
 encrypted\_text = self.basic\_decrypt\_text.toPlainText()  
 key = self.basic\_key.text()  
 decrypted = encrypted\_text[::-1]  
 self.result.setText(f"Дешифровано. Ключ: {key}\nЗашифрованный текст: {encrypted\_text}\nРезультат: {decrypted}")  
  
class AdvancedScreen(ContentFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 lbl = QLabel("Продвинутый режим")  
 lbl.setStyleSheet(f"font-size: 20px; color: {TEXT\_LIGHT}; font-weight: 300;")  
 self.addWidget(lbl)  
  
 enc\_frame = QFrame()  
 enc\_layout = QVBoxLayout(enc\_frame)  
 lbl1 = QLabel("Текст для шифрования:")  
 lbl1.setStyleSheet(f"color: {TEXT\_LIGHT};")  
 enc\_layout.addWidget(lbl1)  
 self.advanced\_encrypt\_text = QTextEdit()  
 self.advanced\_encrypt\_text.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 enc\_layout.addWidget(self.advanced\_encrypt\_text)  
 self.advanced\_algo = QComboBox()  
 self.advanced\_algo.addItems(["Caesar", "AES", "RSA"])  
 self.advanced\_algo.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 enc\_layout.addWidget(self.advanced\_algo)  
 btn\_enc2 = QPushButton("Зашифровать")  
 btn\_enc2.setStyleSheet(f"background: {BORDER}; color: {TEXT\_LIGHT}; border: none; padding: 6px;")  
 btn\_enc2.clicked.connect(self.advanced\_encrypt)  
 enc\_layout.addWidget(btn\_enc2)  
  
 dec\_frame = QFrame()  
 dec\_layout = QVBoxLayout(dec\_frame)  
 lbl2 = QLabel("Зашифрованный текст:")  
 lbl2.setStyleSheet(f"color: {TEXT\_LIGHT};")  
 dec\_layout.addWidget(lbl2)  
 self.advanced\_decrypt\_text = QTextEdit()  
 self.advanced\_decrypt\_text.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 dec\_layout.addWidget(self.advanced\_decrypt\_text)  
 self.advanced\_algo\_dec = QComboBox()  
 self.advanced\_algo\_dec.addItems(["Caesar", "AES", "RSA"])  
 self.advanced\_algo\_dec.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 dec\_layout.addWidget(self.advanced\_algo\_dec)  
 self.advanced\_key = QLineEdit()  
 self.advanced\_key.setPlaceholderText("Введите ключ")  
 self.advanced\_key.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 dec\_layout.addWidget(self.advanced\_key)  
 btn\_dec2 = QPushButton("Дешифровать")  
 btn\_dec2.setStyleSheet(f"background: {BORDER}; color: {TEXT\_LIGHT}; border: none; padding: 6px;")  
 btn\_dec2.clicked.connect(self.advanced\_decrypt)  
 dec\_layout.addWidget(btn\_dec2)  
  
 splitter = QSplitter()  
 splitter.addWidget(enc\_frame)  
 splitter.addWidget(dec\_frame)  
 self.addWidget(splitter)  
  
 self.adv\_result = QTextEdit()  
 self.adv\_result.setReadOnly(True)  
 self.adv\_result.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 self.addWidget(self.adv\_result)  
  
 def advanced\_encrypt(self):  
 text = self.advanced\_encrypt\_text.toPlainText()  
 algo = self.advanced\_algo.currentText()  
 dummy\_key = "key\_456"  
 encrypted = text[::-1]  
 self.adv\_result.setText(f"Зашифровано с помощью {algo}\nКлюч: {dummy\_key}\nРезультат: {encrypted}")  
  
 def advanced\_decrypt(self):  
 encrypted\_text = self.advanced\_decrypt\_text.toPlainText()  
 algo = self.advanced\_algo\_dec.currentText()  
 key = self.advanced\_key.text()  
 decrypted = encrypted\_text[::-1]  
 self.adv\_result.setText(f"Дешифровано {algo} с ключом '{key}'\nЗашифрованный текст: {encrypted\_text}\nРезультат: {decrypted}")  
  
class SettingsScreen(ContentFrame):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 lbl = QLabel("Настройки")  
 lbl.setStyleSheet(f"font-size: 20px; color: {TEXT\_LIGHT}; font-weight: 300;")  
 self.addWidget(lbl)  
 form = QFormLayout()  
 theme\_label = QLabel("Theme:")  
 theme\_label.setStyleSheet(f"color: {TEXT\_LIGHT};")  
 self.theme\_combo = QComboBox()  
 self.theme\_combo.addItems(["Light", "Dark"] )  
 self.theme\_combo.setStyleSheet(f"background: {DARK\_BG}; color: {TEXT\_LIGHT}; border: 1px solid {BORDER};")  
 form.addRow(theme\_label, self.theme\_combo)  
 save\_btn = QPushButton("Сохранить")  
 save\_btn.setStyleSheet(f"background: {BORDER}; color: {TEXT\_LIGHT}; border: none; padding: 6px;")  
 save\_btn.clicked.connect(self.save\_settings)  
 form.addRow(save\_btn)  
 self.addLayout(form)  
  
 def save\_settings(self):  
 msg = QMessageBox(self)  
 msg.setWindowTitle("Успех")  
 msg.setText("Настройки сохранены!")  
 msg.setStyleSheet(f"""  
 QMessageBox {{  
 background: {CARD\_BG};  
 color: {TEXT\_LIGHT};  
 }}  
 QLabel {{  
 color: {TEXT\_LIGHT};  
 font-size: 14px;  
 }}  
 QPushButton {{  
 background: {BORDER};  
 color: {TEXT\_LIGHT};  
 border: none;  
 padding: 6px;  
 min-width: 80px;  
 }}  
 """)  
 msg.exec\_()  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle("Veil")  
 self.setGeometry(100, 100, 1200, 800)  
 self.setStyleSheet(f"QMainWindow {{ background: {DARK\_BG}; color: {TEXT\_LIGHT}; }}")  
  
 QFontDatabase.addApplicationFont("Roboto-Regular.ttf")  
 self.setFont(QFont("Roboto", 10))  
  
 main = QWidget()  
 self.setCentralWidget(main)  
 hbox = QHBoxLayout(main)  
 hbox.setContentsMargins(0, 0, 0, 0)  
  
 sidebar = QFrame()  
 sidebar.setFixedWidth(80)  
 sidebar.setStyleSheet(f"background: {CARD\_BG}; border-right: 1px solid {BORDER};")  
 vbar = QVBoxLayout(sidebar)  
 vbar.setAlignment(Qt.AlignTop)  
  
 icons = [  
 ("icons/home.svg", "Home"),  
 ("icons/user.svg", "Basic Mode"),  
 ("icons/advanced.svg", "Advanced Mode"),  
 ("icons/settings.svg", "Settings"),  
 ]  
 for idx, (icon, tip) in enumerate(icons):  
 btn = SideBarButton(icon, tip)  
 btn.clicked.connect(lambda \_, i=idx: self.switch\_page(i))  
 vbar.addWidget(btn)  
  
 hbox.addWidget(sidebar)  
  
 wrapper = QWidget()  
 vbox = QVBoxLayout(wrapper)  
 vbox.setContentsMargins(20, 20, 20, 20)  
  
 self.title\_bar = TitleBar()  
 vbox.addWidget(self.title\_bar)  
  
 self.stack = AnimatedStackedWidget()  
 self.stack.addWidget(HomeScreen())  
 self.stack.addWidget(BasicScreen())  
 self.stack.addWidget(AdvancedScreen())  
 self.stack.addWidget(SettingsScreen())  
 vbox.addWidget(self.stack)  
  
 hbox.addWidget(wrapper)  
  
 def switch\_page(self, idx):  
 pages = ["Home", "Basic", "Advanced", "Settings"]  
 self.title\_bar.title.setText(pages[idx])  
 self.stack.setCurrentIndex(idx)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 window = MainWindow()  
 window.show()  
 sys.exit(app.exec\_())