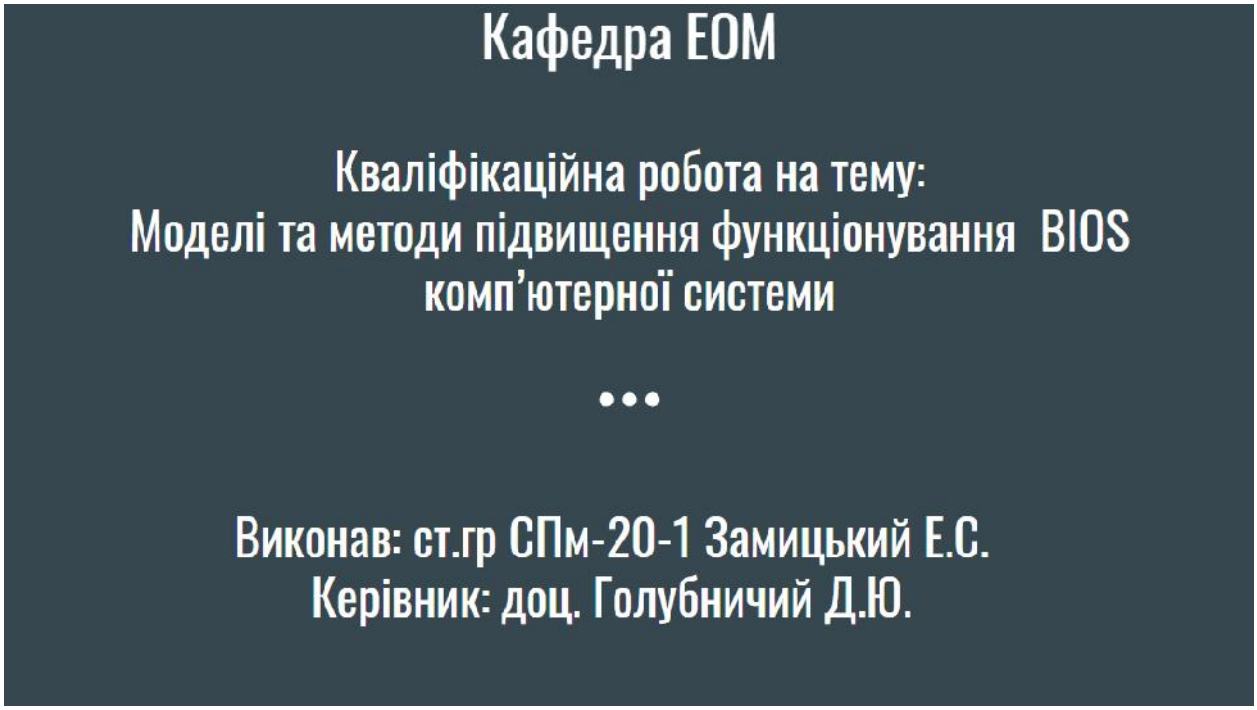
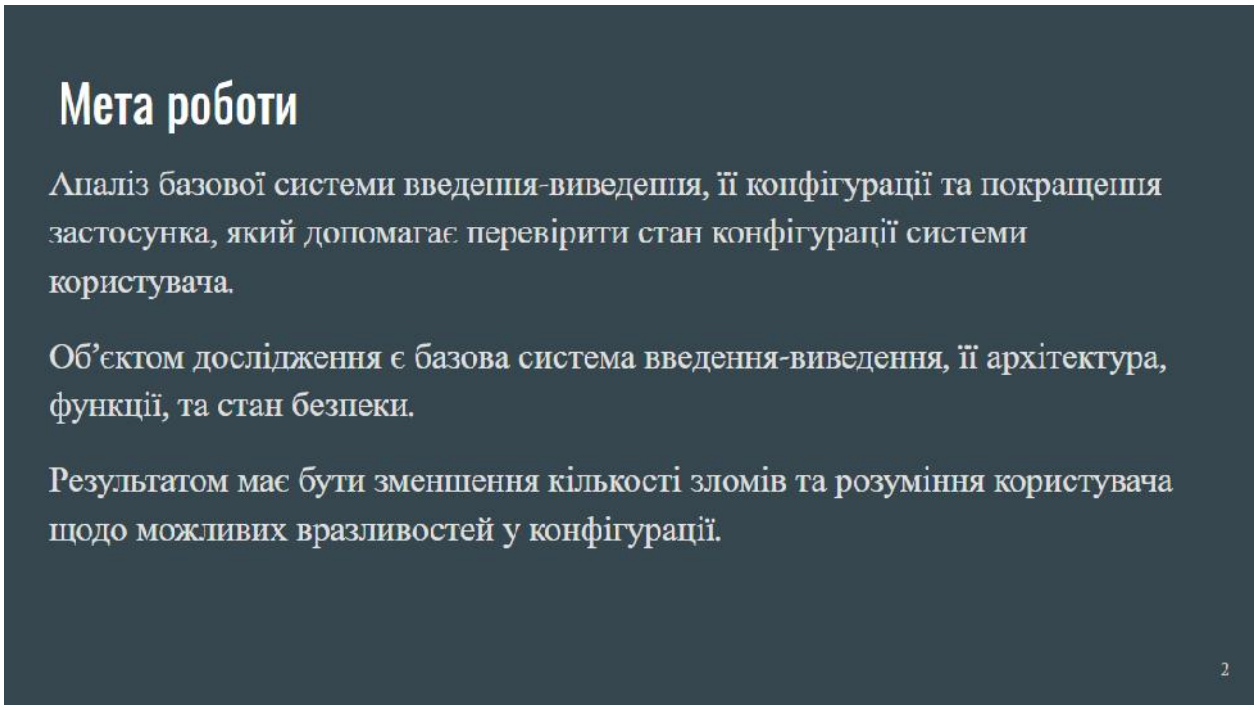
ДОДАТОК А

Графічний матеріал кваліфікаційної роботи





Аналіз проблеми

3

Аналіз проблеми

BIOS комп'ютери	Оновлення	Конфігурація
• Невикористання UEFI при наявності • "Комбінований" режим • Найбільша вірогідність злому	• Регулярність оновлень • Цілісність	• Fast/Quick boot • Secure boot • Пароль для адміністрування системою

4

National Vulnerability Database

2021

На даний момент було виявлено 11 зломів UEFI системи, які включають:

- Adjacent access
- Buffer overflow
- Certificate validation
- Boot logic

2020

Стосовно минулого року:

- Insufficient authenticity check
- Overwrite vulnerability
- Callback functions

5

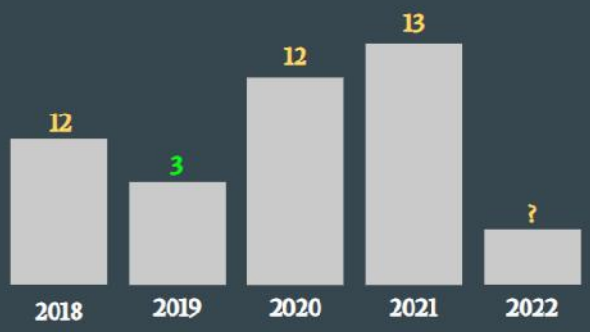
BIOS

Результати статистики зломів за останні роки

6

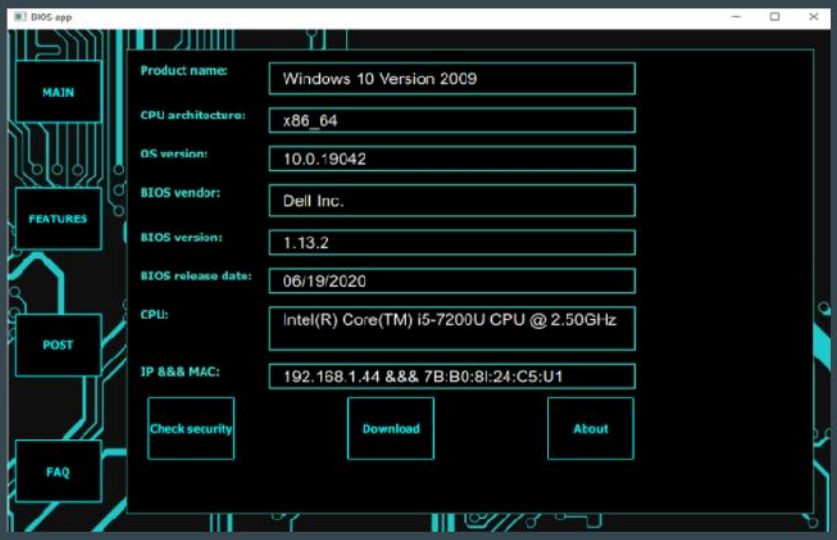
UEFI

Результати статистики зломів за останні роки



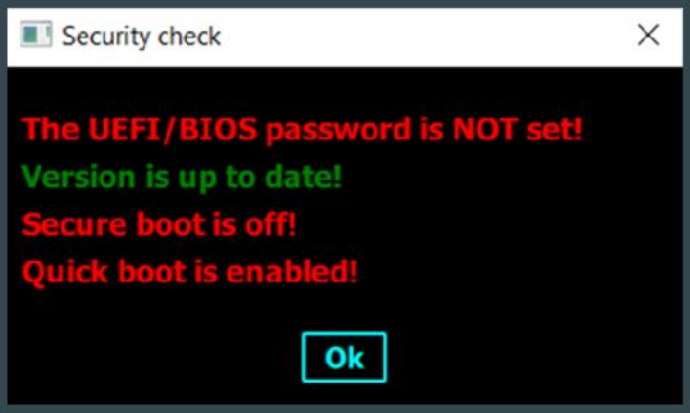
7

Програмний додаток



8

Перевірка конфігурації на рівні ОС



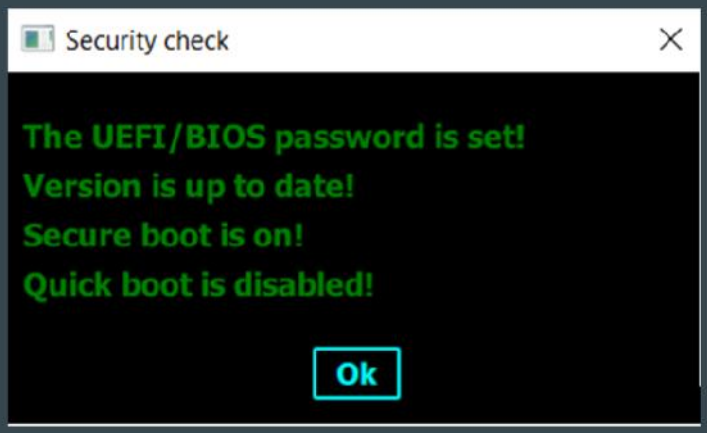
9

Очікувана конфігурація користувача

Secure boot	• Увімкнений за наявності підтримки
Fast boot	• Неможливий у випадку “дуо-систем” • Вимкнений
Оновлення	• Оповіщення користувача про нове оновлення • Регулярні оновлення
Пароль	• Наявність строгого паролю до налаштувань конфігурації

10

Очікувані результати перевірки конфігурації



11

Висновки

В результаті виконання кваліфікаційної роботи було проведено аналіз методів управління комп'ютерних систем BIOS та UEFI, який включав у себе принципи роботи обох систем, різниця між системами. На основі недоліків одної з систем був описан перехід від цієї системи до більш нової, покращення налаштувань конфігурацій даних систем, та оновлення програмного застосунку задля допомоги звичайному користувачеві перевірки найважливіших налаштувань зі сторони кібербезпеки.

12

ДОДАТОК Б

Програмний код

Б.1 Отримання системної інформації

Б.1.1 Реалізація класу Sysinfo

```
include "sysinfo.h"


SysInfo::SysInfo(QObject *parent)
    :QObject(parent), infoObject(),
settings("HKEY_LOCAL_MACHINE\\HARDWARE\\DESCRIPTION\\System\\BIOS",
QSettings::NativeFormat)
{

}

QString SysInfo::cpuArchitecture() const
{
    return
QString::fromStdString(std::string(infoObject.currentCpuArchitecture().toLoca
l8Bit().constData()));
}

QString SysInfo::productType() const
{
    return
QString::fromStdString(std::string(infoObject.prettyProductName().toLocal8Bit
().constData()));
}

QString SysInfo::kernelVersion() const
{
     return
QString::fromStdString(std::string(infoObject.kernelVersion().toLocal8Bit().c
onstData()));
}

QString SysInfo::productName() const
{
    return
QString::fromStdString(std::string(settings.value("SystemProductName",
"0").toString().toLocal8Bit().constData()));
}

QString SysInfo::biosVersion() const
{
    return QString::fromStdString(std::string(settings.value("BIOSVersion",
"0").toString().toLocal8Bit().constData()));
}

QString SysInfo::biosVendor() const
{
```

```
    return QString::fromStdString(std::string(settings.value("BIOSVendor",
"0").toString().toLocal8Bit().constData()));
}

QString SysInfo::biosReleaseDate() const
{
    return
QString::fromStdString(std::string(settings.value("BIOSReleaseDate",
"0").toString().toLocal8Bit().constData()));
}

QString SysInfo::network() const
{
    foreach(QNetworkInterface networkInterface,
QNetworkInterface::allInterfaces())
        {
            if (networkInterface.flags().testFlag(QNetworkInterface::IsUp) &&
!networkInterface.flags().testFlag(QNetworkInterface::IsLoopBack))
            {
                foreach (QNetworkAddressEntry entry,
networkInterface.addressEntries())
                {
                    if ( entry.ip().toString().contains(".")){

                        return
QString::fromStdString(std::string(entry.ip().toString().toLocal8Bit().constD
ata()) +
                                   " &&& "
                                   +
std::string(networkInterface.hardwareAddress().toLocal8Bit().constData()));}}
}}}

QString SysInfo::cpuInfo() const
{
    QProcess process_system;
    QString cpu_output;

            if(QSysInfo::kernelType() == "winnt")
            {
                QString cpuname = "wmic cpu get name";
                process_system.start(cpuname);
                process_system.waitForFinished();

                cpu_output = process_system.readLine();
                cpu_output = process_system.readLine();

            }
            return
QString::fromStdString(std::string(cpu_output.toLocal8Bit().constData()));
}
```

Б.2 Опис інтерфейсу застосунка

Б.2.1 Реалізація фрагменту Main

```
MyFrame {
    id: myf

    Flickable {
```

```
        anchors.fill: parent
        contentHeight: controlsColumn.height
        clip: true
        bottomMargin: 50

            Grid {
                id: controlsColumn
                anchors{left: parent.left; right: parent.right; leftMargin:
20}
                columns: 2
                spacing: 20
                topPadding: 20
                MyLabel {text: "Product name: ";}

                Rectangle {
                    width: 550
                    height: myText.contentHeight + 20
                    color: "transparent"
                    border {color: "turquoise"; width: 2}

            Text {
                id: myText
                anchors.fill:parent
                font {family: "Helvetica"; pointSize: 15;}
                text:  sys.productType
                color: "white"
                leftPadding: 20
                horizontalAlignment: Text.left
                verticalAlignment: Text.AlignVCenter
            }
        }

        MyLabel {text: "CPU architecture: " ;}

        Rectangle {
            width: 550
            height: myText1.contentHeight + 10
            color: "transparent"
            border {color: "turquoise"; width: 2}

            Text {
                id: myText1
                anchors.fill:parent
                font {family: "Helveticeca"; pointSize: 15}
                text: sys.cpuArchitecture
                color: "white"
                leftPadding: 20
                horizontalAlignment: Text.left
                verticalAlignment: Text.AlignVCenter
            }
        }

        MyLabel {text: "OS version: " ;}
        Rectangle {
            width: 550
            height: myText2.contentHeight + 10
            color: "transparent"
            border {color: "turquoise"; width: 2}

            Text {
                id: myText2
                anchors.fill:parent
                font {family: "Helvetica"; pointSize: 15}
                text:  sys.kernelVersion
```

```
        color: "white"
        leftPadding: 20
        horizontalAlignment: Text.left
        verticalAlignment: Text.AlignVCenter
    }
}

MyLabel {text: "BIOS vendor: " ; }
Rectangle {
    width: 550
    height: myText3.contentHeight + 20
    color: "transparent"
    border {color: "turquoise"; width: 2}

    Text {
        id: myText3
        anchors.fill:parent
        font {family:"Helvetica"; pointSize: 15}
        text:  sys.biosVendor
        color: "white"
        leftPadding: 20
        horizontalAlignment: Text.left
        verticalAlignment: Text.AlignVCenter
    }
}

MyLabel {text: "BIOS version: " ;}
Rectangle {
    width: 550
    height: myText4.contentHeight + 10
    color: "transparent"
    border {color: "turquoise"; width: 2}

    Text {
        id: myText4
        anchors.fill:parent
        font {family:"Helvetica"; pointSize: 15}
        text:  sys.biosVersion
        color: "white"
        leftPadding: 20
        horizontalAlignment: Text.left
        verticalAlignment: Text.AlignVCenter
    }
}

MyLabel {text: "BIOS release date: " ;}
Rectangle {
    width: 550
    height: myText5.contentHeight + 10
    color: "transparent"
    border {color: "turquoise"; width: 2}

    Text {
        id: myText5
        anchors.fill:parent
        font {family:"Helvetica"; pointSize: 15}
        text:  sys.biosReleaseDate
        color: "white"
        leftPadding: 20
        horizontalAlignment: Text.left
        verticalAlignment: Text.AlignVCenter
    }
}
```

```
        MyLabel {text: "CPU: " ;}
        Rectangle {
            width: 550
            height: myText6.contentHeight + 10
            color: "transparent"
            border {color: "turquoise"; width: 2}

            Text {
                id: myText6
                anchors.fill:parent
                font {family:"Helvetica"; pointSize: 15}
                text:  sys.cpuInfo
                color: "white"
                leftPadding: 20
                horizontalAlignment: Text.left
                verticalAlignment: Text.AlignVCenter
            }
        }
        MyLabel {text: "IP &&& MAC: " ;}
        Rectangle {
            width: 550
            height: myText7.contentHeight + 10
            color: "transparent"
            border {color: "turquoise"; width: 2}

            Text {
                anchors.fill:parent
                id: myText7
                font {family:"Helvetica"; pointSize: 15}
                text:  sys.network
                color: "white"
                leftPadding: 20
                horizontalAlignment: Text.left
                verticalAlignment: Text.AlignVCenter
            }}} }
Row {
    id: main_row
    x: 32
    y: 524

    spacing: 170
    MyButton {
        id: _btn_biosCheck
        text: "Check BIOS"
        //button_ma.onClicked: msg_dial.open()
        button_ma.onClicked: {
            var component = Qt.createComponent("Popup.qml");
                    if (component.status === Component.Ready) {
                    var dialog = component.createObject(parent,{popupType:
1});
                    dialogConnection.target = dialog
                    dialog.show();}

    }
    }
    Connections {
        id: dialogConnection
        onVisibleChanged: {
            if(!target.visible)
                console.log(target.returnValue);
        }
    }

    MyButton {text: "Download"}
```

```
    MyButton {text: "About"}

    }
}
```

Б.2.2 Реалізація фрагменту Features

```
MyFrame {
    Flickable{
        anchors.fill: parent
        contentHeight: controlsColumn.height
        clip: true
        Column{
            id: controlsColumn
            spacing: 10
            anchors{
                left: parent.left
                right: parent.right
                margins: 5
            }
              YourItem{ text: "+ Standard CMOS Features"; /*item_color:
"turquoise";*/ source: "Features_standard.qml"}
              YourItem{ text: "+ Advanced BIOS Features"; /*item_color:
"turquoise";*/ source: "Features_advanced.qml"}
              YourItem{ text: "+ Integrated Peripherals"; /*item_color:
"turquoise";*/ source: "Features_peripherals.qml"}
              YourItem{ text: "+ Power Management Setup"; /*item_color:
"turquoise";*/ source: "Features_powerSetup.qml"}
              YourItem{ text: "+ PnP/PCI Configurations"; /*item_color:
"turquoise";*/ source: "Features_pci.qml"}
              YourItem{ text: "+ PC Health Status"; /*item_color:
"turquoise";*/ source: "Features_health.qml"}
              YourItem{ text: "+ Frequency/Voltage Control"; /*item_color:
"turquoise";*/ source: "Features_voltage.qml";}
        }
        ScrollBar.vertical: ScrollBar{}
    }
}
```

Б.2.3 Реалізація фрагменту POST

```
MyFrame {
    id: my_frame

    Flickable{
        anchors.fill: parent
        contentHeight: controlsColumn.height
        clip: true

        Column{
            id: controlsColumn
            width: parent.width
            spacing: 10

            Text {
                id: txt
                //anchors {fill: parent; topMargin: 10;}
                color: "white"
```

```
                text: "<u>AMI BIOS BEEP CODES</u> <br><br>\nBelow are the AMI
BIOS beep codes.
However, because of the wide variety of different computer manufacturers with
this BIOS, the beep codes may vary."
                wrapMode: Text.WordWrap
                font {bold: true; pixelSize: 15; }
                height: Text.paintedHeight
                leftPadding: 50
                topPadding: 20
                bottomPadding: 20
            }

        MyTableView {
            id: tableView
            width: my_frame.width - 50
            height: 888

            model: ListModel {
                id: listModel

                ListElement {beep: "1 short"; description: "DRAM refresh
failure"}
                ListElement {beep: "2 short"; description: "Parity circuit
failure"}
                ListElement {beep: "3 short"; description: "Base 64K RAM
failure"}
                ListElement {beep: "4 short"; description: "System timer
failure"}
                ListElement {beep: "5 short"; description: "Process failure"}
                ListElement {beep: "6 short"; description: "Keyboard
controller Gate A20 error"}
                ListElement {beep: "7 short"; description: "Virtual mode
exception error"}
                ListElement {beep: "8 short"; description: "Display memory
Read/Write test failure"}
                ListElement {beep: "9 short"; description: "ROM BIOS checksum
failure"}
                ListElement {beep: "10 short"; description: "CMOS shutdown
Read/Write error"}
                ListElement {beep: "11 short"; description: "Cache memory
error"}
                ListElement {beep: "1 long, 3 short"; description:
"Conventional/Extended memory failure"}
                ListElement {beep: "1 long, 8 short"; description:
"Display/Retrace test failed"}
                ListElement {beep: "Two-tone siren";  description: "Low CPU
fan speed, voltage level issue"}
            }

        }

        Text {
            id: txt0
            text: "<u>Award BIOS beep codes</u> <br><br>
                    Below are Award BIOS beep codes. However, because of the
wide variety of different
                    computer manufacturers with this BIOS, the beep codes may
vary."
            color: "white"
            wrapMode: Text.WordWrap
            font {bold: true; pixelSize: 15; }
            leftPadding: 50
            topPadding: 50
            bottomPadding: 20
```

```
        }

        MyTableView {
            id: tableView1
            height: 470
            width: my_frame.width - 50
            model: ListModel {
            id: listModel1
            //DELL beep codes
            ListElement {beep: "1 beep"; description: "BIOS ROM corruption or
failure"}
            ListElement {beep: "2 beeps"; description: "Memory (RAM) not
detected"}
            ListElement {beep: "3 beeps"; description: "Motherboard failure"}
            ListElement {beep: "4 beeps"; description: "Memory (RAM)
failure"}
            ListElement {beep: "5 beeps"; description: "CMOS battery
failure"}
            ListElement {beep: "6 beeps"; description: "Video card failure"}
            ListElement {beep: "7 beeps"; description: "Bad processor (CPU)"}
            }
           }

        Text {
            id: txt1
            text: "<u>DELL BEEP CODES</u> <br><br>"
            wrapMode: Text.WordWrap
            color: "white"
            font { bold: true; pixelSize: 15; }
            leftPadding: 50
            topPadding: 50
            bottomPadding: 20
        }

        MyTableView {
            id: tableView2
            height: 650
            width: my_frame.width - 50
            model: ListModel {
            id: listModel2
            //IBM BIOS beep codes
            ListElement {beep: "No beeps"; description: "No power, loose
card, or short"}
            ListElement {beep: "1 short"; description: "Normal POST, computer
is ok"}
            ListElement {beep: "2 short"; description: "POST error, review
screen for error code"}
            ListElement {beep: "Continuous beep"; description: "No power,
loose card, or short"}
            ListElement {beep: "1 repeating short"; description: "No power,
loose card, or short"}
            ListElement {beep: "1 long & 1 short"; description: "Motherboard
issue"}
            ListElement {beep: "1 long & 2 short"; description: "Video
(Mono/CGA display circuitry) issue"}
            ListElement {beep: "1 long & 3 short"; description: "Video (EGA)
display circuitry"}
            ListElement {beep: "3 long"; description: "Keyboard or keyboard
card error"}
            ListElement {beep: "Blank display"; description: "Video display
circuitry"}
                }
            }
```

```
        Text {
            id: txt2
            text: "<u>IBM BIOS beep codes</u> <br><br>
                  Below are general IBM BIOS beep codes. However, because of
the wide
                  variety of models shipping with this BIOS, the beep codes
may vary."
            color: "white"
            wrapMode: Text.WordWrap
            font { bold: true; pixelSize: 15; }
            leftPadding: 50
            topPadding: 50
            bottomPadding: 20
        }

        MyTableView {
            id: tableView3
            width: my_frame.width - 50
            height: 350
            model: ListModel {
            id: listModel3
            ListElement {beep: "1 long, 2 short"; description: "Indicates a
video error has occurred and the BIOS cannot initialize the video screen to
display any additional information"}
            ListElement {beep: "1 long, 3 short"; description: "Video card
not detected (reseat video card) or bad video card"}
            ListElement {beep: "Repeating endlessly."; description: "RAM
problem"}
            ListElement {beep: "Repeated high frequency beeps while PC is
running"; description: "Overheating processor (CPU)"}
            ListElement {beep: "Repeated beeps alternating high & low
frequency"; description: "Issue with the processor (CPU), possibly damaged"}
    }
}

        Text {
            id: txt3
            text: "AMI BIOS BEEP CODES"
            color: "white"
            font {underline: true; bold: true; pixelSize: 15; }
            leftPadding: 50
            topPadding: 50
            bottomPadding: 20
         }
       }
        ScrollBar.vertical: ScrollBar{}
    }}
```

Б.2.4 Реалізація фрагменту FAQ

```
MyFrame {
    Flickable{
        anchors.fill: parent
        contentHeight: controlsColumn.height
        clip: true
        Column{
            id: controlsColumn
            spacing: 10
            anchors{
                left: parent.left
                right: parent.right
```

```
                margins: 5
            }
                YourItem{ text: "Как зайти в BIOS?"; source:
"Q1.qml"}
                YourItem{ text: "Как сбросить BIOS?"; source:
"Q2.qml"}
                YourItem{ text: "Не работает клавиатура при
запуске компьютера?"; source: "Q3.qml"}
                YourItem{ text: "Не запускается Windows?"; source:
"Q4.qml"}
                YourItem{ text: "Постоянно сбивается время на
компьютере?"; source: "Q5.qml"}
                YourItem{ text: "Компьютер не видит мышь?";
source: "Q6.qml"}
                YourItem{ text: "Жесткий диск издает звуки?";
source: "Q7.qml"}
            }
            ScrollBar.vertical: ScrollBar{}
    }}
```