Final Project

Student mental health test

based on Beck's Depression Inventory

Boston University College of Engineering

EC327 Introduction to Software Engineering, Summer 2022

Submitted by Team 10,

Wenhao Cao, Project lead & Specification Lead, wenhaoc@bu.edu

Youchen Li, Interface Lead, youchenli@bu.edu

Heshan Chen, Technical Lead, ephsc@bu.edu

Wenhao Cao 33.3%, Youchen Li 33.3%, Heshan Chen 33.3%.

Project GitHub repository: https://github.com/youcc65/EC327-project-team-10
Project Video URL: https://youtu.be/HPubOOxiopo

Overall Software Architecture *by Wenhao Cao*

21 questions from Beck's test; in .txt format, cited by C++

Description of each component

Front-end *by Youchen Li*:

The Front-end of this project requires explicit texts to show the users how to use this application as efficiently as possible. Multiple graphical interfaces are created by codes in C++. The contents below demonstrate the coding of the Front-end of the application.

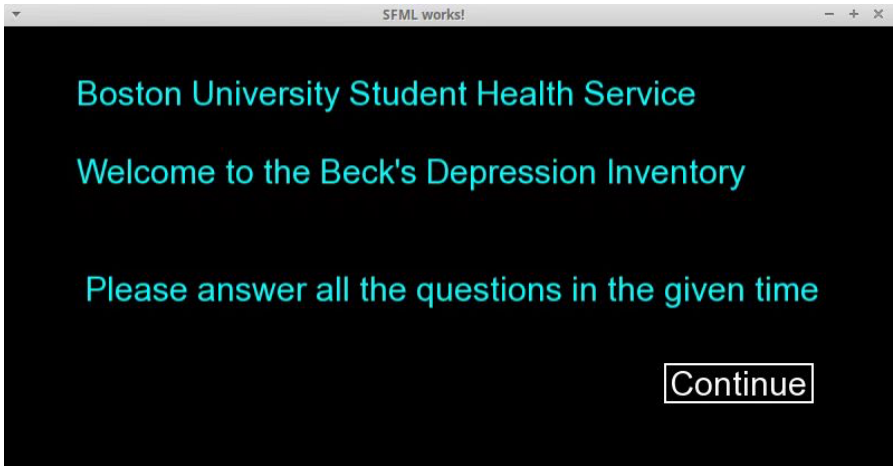


Fig 1.1 The Menu

```
void initMenu(std::string str)
{
    text.setString(str);
    text.setFont(font);
    text.setCharacterSize(30);
    text.setFillColor(sf::Color::Cyan);

    // center text
    sf::FloatRect textRect = text.getLocalBounds();
    text.setOrigin(textRect.left + textRect.width / 2.0f,
                   textRect.top + textRect.height / 2.0f);
    sf::Vector2f screenCenter = window.getView().getCenter();
    text.setPosition(screenCenter.x, 150);
}
```

Fig 1.1.2 The Setting of the Menu

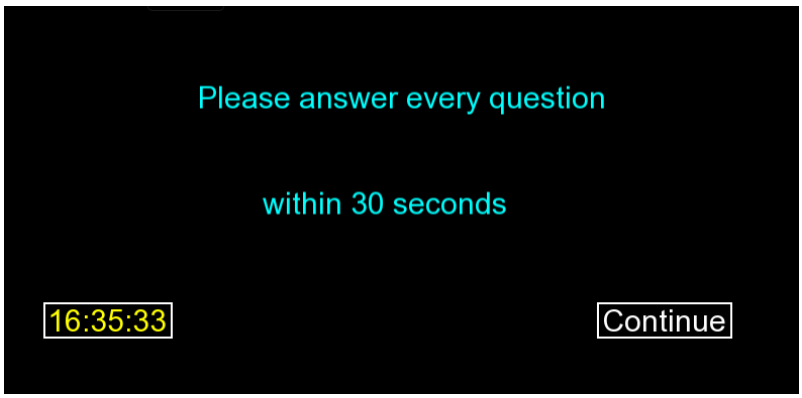


Fig 1.2 Local time

To set the Menu of the application, texts need to be assigned with the correct font, size, and color. It also needs to be positioned in the center of the screen. There are also texts without borders and texts with borders, such as the NEXT button, so we need to initialize the text with and without the border by setting its position, font, color, and size.

In figure 1.2, we need to display the user's local time using texts with border and display it on the screen with yellow color.

```
void showTime()
{
    // get the time now
    time_t now;
    time(&now);
    char timestr[128];
    strftime(timestr, 128, "%H:%M:%S", localtime(&now));

    // display the time
    timeText.setCharacterSize(30);
    initTextWithBorder(timeText, timeBorder, timestr, 50, 300, 30,
sf::Color::Yellow, sf::Color::White);
    window.draw(timeBorder);
    window.draw(timeText);
}
```

Fig 1.2.1 Get time & display the time

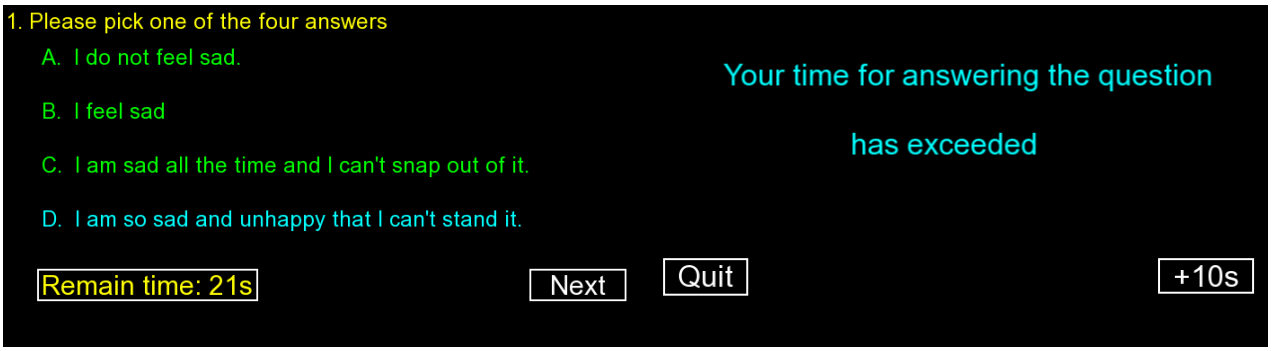


Fig 1.3 Question page & Fig 1.4 Time exceeded page

In Figures 1.3 and 1.4, the user must answer every question in 30 seconds. Otherwise, the application will pop up on a page in which the user has the opportunity to answer it in ten

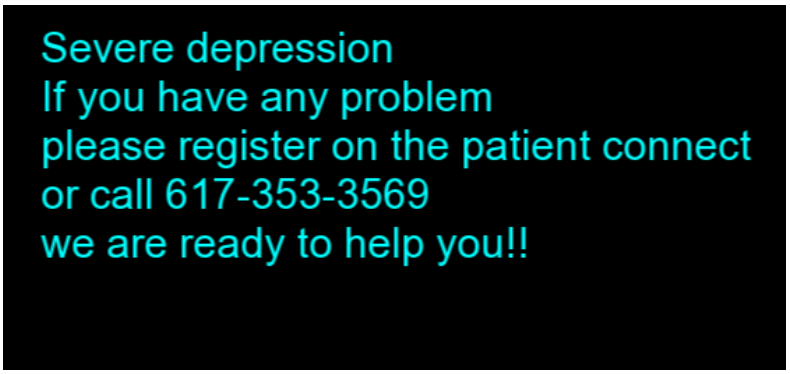
more seconds or quit. Once the user clicks on the answer they want, the color turns from green to cray.

```
// select a qustion by its index and set its color
void selectQuestion(int id)
{
    for (int j = 0; j < 4; j++)
    {
        textQustions[j].setFillColor(sf::Color::Green);
    }
    textQustions[id].setFillColor(sf::Color::Cyan);
    questionSelected = id;
}
```

Fig 1.3.1 Question page reset answer's color if selected

```
void init()
{
    initMenu("Your time for answering the question\n\n\t\t\t\thas exceeded");
    quit.init(" Quit ", 100, 300);
    extendTime.init(" +10s ", 600, 300);
}
```

Fig 1.4.1 Time exceeded page



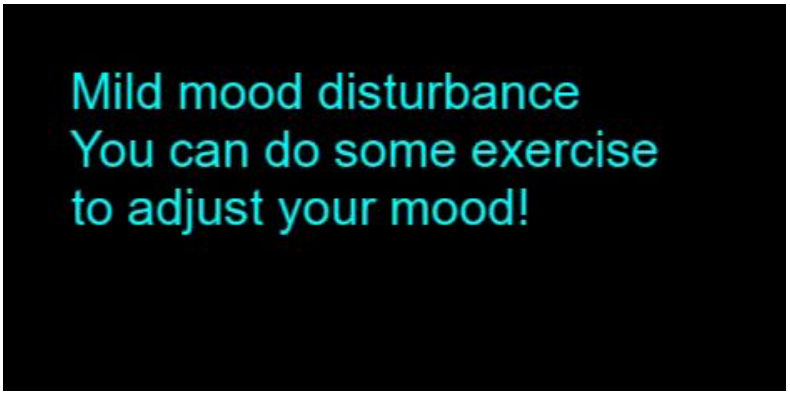


Fig 1.5 Test Result examples (2 of 7)

In figure 1.5, the result of the test will appear on the screen after the user finish all the questions in the given time.

Coding functions or string required for the Front-end:

Function or string name	Purpose
void **intiMenu**	Intialize the menu, which has centered text
void **initText**	Initialize a text without border by setting its position, color, size, and font
void **initTextWithBorder**	Intialize a text with border by setting its position, color, size, and font
void **showTime**	Get the local time and display it
class **ClockCountingDown**	Displays the counting clock
class **TextWithoutBorder**	Constructs text without border
class **TextWithBorder**	Constructs text with border
void **showMenu**	Show the menu in the window
class **QuestionFrame**	Initialize a question frame, showing the questions to select
void **selectQuestion**	Select an answer by its index and set its color
string **summaries**	Display all the possible test results

Back-end *by Heshan Chen* :

As a questionnaire for users, most of our program's code is front-end. What the user doesn't see includes the scoring and judgment logic after making a choice. Our program uses the scoring rules and judgment conditions that match Beck's Depression Inventory. When the user selects among the four options, the points for each corresponding question will accumulate from 0. The first option represents zero points, and the second option represents one point. The third option represents two, while the fourth option represents three points.

```
0 I do not feel like a failure.
1 I feel I have failed more than the average person.
2 As I look back on my life, all I can see is a lot of failures.
3 I feel I am a complete failure as a person.
```

Fig 1.6 Test question example

As shown in the figure, we divided the total score of 63 points into seven intervals, each corresponding to a different test result. The program returns a result of 0 when the user's total score is 0. The program returns the first result when the user's total score is between 1 and 10. The program returns the second result when the user's total score is between 11 and 16. The program returns the third result when the user's total score is between 17 and 20. The program returns the fourth result when the user's total score is between 21 and 30. The program returns the fifth result when the user's total score is between 31 and 40. The program returns the sixth result when the user's total score exceeds 40.

```
std::string getSummaryString(int score)
{
    std::string summaries[] = {
        "You are very healthy \n Enjoy your life!!!!!!1",
        "These ups and downs are considered normal \n Have a good day!!!!!!!",
        "Mild mood disturbance \nYou can do some exercise\nto adjust your mood!",
        "Borderline clinical depression\nIf you have any problem\nplease register on the patient connect\nwe are ready to help you!!",
        "Moderate depression \nIf you have any problem\nplease register on the patient connect\nwe are ready to help you!!",
        "Severe depression \nIf you have any problem\nplease register on the patient connect\nor call 617-353-3569\nwe are ready to help you!!",
        "Extreme depression  \nIf you have any problem\nplease register on the patient connect\nor call 617-353-3569\nwe are ready to help you!!"};

    if (score == 0)
    {
        return summaries[0];
    }
    else if (between(score, 1, 10))
    {
        return summaries[1];
    }
    else if (between(score, 11, 16))
    {
        return summaries[2];
    }
    else if (between(score, 17, 20))
    {
        return summaries[3];
    }
    else if (between(score, 21, 30))
    {
        return summaries[4];
    }
    else if (between(score, 31, 40))
    {
        return summaries[5];
    }
    else if (score > 40)
    {
        return summaries[6];
```

Fig 1.7 Grading logic &

```
int curPage = 0;
bool pageChanged = true;
QuestionFrame qFrame;
TimeExceedFrame tFrame;
ClockCountingDown clk;
enum STATUS
{
    NOTSTART,
    NORMAL,
    EXCEED,
    FAILED,
    EXTENDED
};
int status = NOTSTART;
int totalScore = 0;


int main()
{
    if (!font.loadFromFile("./arial.ttf"))
    {
        std::cerr << "Error loading font file!" << std::endl;
        return -1;
    }

    loadQuestionsFromFile("questions.txt");
```

Fig 1.8 6 STATUS

The program also needs to have import font, title, and other functions. At the same time, the back-end program also needs to ensure that the user can change options during the process. If the user does not choose within 30 seconds, the application should automatically exit or ask the user if they need more time. Therefore, to deal with a wide variety of possible situations, we defined "STATUS" and divided the program into NOTSTART, NORMAL, EXCEED, FAILED, and EXTENDED. NOTSTARD means that the user hasn't started to answer the question, so the user's total score is zero. EXCEED represents a timeout for an answer, EXTENDED represents an extension of 10 seconds for an answer and FAILED is only called when an extension of 10 seconds for an answer expires again. The program will exit automatically.

Target user and marketability

According to the Mayo Clinic, up to 44% of college students reported having symptoms of depression and anxiety, including difficulty handling schoolwork, loss of interest in activities, changes in eating or sleeping patterns, emotional outbursts, panic, lack of energy, etc. These

problems seriously affect college students' mental health, school life, and academic ability. Due to the restrictions on the treatments to the clinic depression, rarely existed productive medicines or specific plans to cure the disease. Therefore, the great importance of primary detection has been gradually appealed, as the school must investigate students' psychological states and provide necessary help to students with mental health problems. Because of the urgent need, we tend to develop an app that can preliminarily diagnose students' mental health status. Based on the now universally accepted Beck's Depression Inventory, our application can initially diagnose students' mental health through twenty-one professional questions, revealing the mental status and providing the executable options. **This App can not only achieve commercial success, but also reflect BU social engineers' love and responsibility for the community.**

Work Cited

https://www.mayoclinichealthsystem.org/hometown-health/speaking-of-health/college-students-and-depression