The Student Sentiment Sensor (SSS)

Team Members: Jack Carson, Dave Pleteau, John Privitera, Mark Morton, Tyler Ceballo

Advisor: Professor Taskin Padir

Abstract

College students may be plagued with many stressors during their academic terms. These stressors, if not handled, can lead to mental health symptoms such as anxiety and depression. Thankfully: schools offer counseling, external medical support exists, and most students have at least a friend or family member willing to support them. However, many students do not seek help for reasons like aversion to painful subjects, a general unawareness of available solutions, or lack of time to carefully search for it. In such scenarios, students may only receive help if somebody starts the conversation with them. Using the Student Sentiment Sensor (SSS), we hope to proactively provide stress-mitigation resources in hopes of improving student mental well being.

This project attempts to mitigate stress by engaging with students through a series of survey-like questions that narrow down their possible stressors and provide relevant assistive resources. The SSS begins its operation by camera-scanning its field of view for students. While scanning, the device uses the OpenCV and Facial Expression Recognition Python library to identify any possibly distressed students by observing their facial expressions. If a possibly distressed student is identified, the device uses the Python library to approximate the negative emotion they're likely feeling. If the targeted student decides to engage with the device, they are informed of what their feeling was approximated to be. Subsequently, the device leverages a decision tree—tailored using results from academic papers and surveys exploring college stressors—to guide the student to a final solution page containing advice and helpful resources. A spreadsheet containing questions, responses, solutions, and resources is used to record visited decision tree nodes. For the convenience of the student, the device offers to email the advice and resources given. Finally, the device provides a revised emotion approximation (calculated based on their decision tree choices) and offers an optional survey that requests feedback on its usefulness and accuracy. The hardware components for this project were a Pi Camera Module 2, Raspberry Pi 7" Touch Screen Display, and Raspberry Pi 4 Model B.

Data collected by this device includes the emotion detected by the computer vision library, the order of questions and responses chosen by the user, whether or not the user completes the tree, and if an email was sent. With this data, the SSS team is able to see which problems are most commonly affecting students. This data will also be used as feedback for potential future iterations of the project.

1. Introduction

1.1 Motivation

You don't want to be stressed out for a long time. The human body, though its "fight or flight" response is useful in dire circumstances, is simply not equipped to handle the effects of excess mental strain. Study after study has unmasked the health consequences of chronic stress, some comparing it to the likes of smoking 5 additional cigarettes per day [1]. Specifically, continued emotional strain can lead to serious health effects such as heart disease, high blood pressure, and diabetes [2]. Unfortunately, millions of US college students are overly stressed out. Not only do college-goers push their limits to complete exceptionally busy workloads, but their developing minds can become conditioned to habitually act from distress [3] causing them to collapse into anxiety, depression, memory-deficiency, and other mental illnesses as well.

Many college students do not properly manage their stress [4] and—despite all there is on the internet—often don't have the mindfulness or energy to seek out information they need. Primary solutions to mental illness such as professional help and lifestyle changes are available but are not always promoted thoroughly, not to mention with regard to a person's specific situation. Online resources present an opportunity to improve students' mental health, but lack an ability to proactively reach out to individuals who exhibit signs of distress.

1.2 Goals

Our goal for this project was to pursue students who appear emotionally unwell and provide them with helpful information and next steps based on their emotional state. This involves sharing free and Northeastern affiliated resources such as tutoring or counseling. To track the effectiveness of this project, we recorded where as well as how far users traverse through the device's dialog and we gathered feedback from users through a survey.

More concretely, we wanted to create a camera-and-touchscreen-based device that tracks students, including those in motion, across its field of view using person tracking. We leveraged a facial emotion detection computer vision (CV) model in order to calculate emotional negativity of candidates close enough to the device, and engaged the most distressed in a conversation relating to their troubles. Once their emotions and/or cause of stress were narrowed down, we presented a rudimentary course of action through use of digital resources.

2. Design

2.1 Hardware

Raspberry Pi

The Raspberry Pi 4 model B was used as the central hardware component of the SSS. The device was chosen for its high computational-ability-to-size ratio. The quad-core Cortex-A72 processor was capable of running computer vision emotional detection on a person in a matter of 6 seconds maximum, and the physical device could be mounted on a consumer phone stand for ease of use and presentation.

Camera

The Pi Camera Module 2 was chosen as the computer vision input device because of its intrinsic compatibility with the Raspberry Pi device. At 8 megapixels, its resolution was more than high enough to perform all of the desired computer vision tasks. The device is capable of capturing video at 1080p, meaning that image quality would not be a limiting factor if real-time video capture and analysis is attempted in the future.

Touchscreen

The Raspberry Pi 7" Touch Screen Display was also chosen for its compatibility with the Raspberry Pi. The display reaches a resolution of 800 x 480 pixels, which we deemed high enough to accurately display the frontend. Additionally, the screen only weighs about 75 grams, allowing it to be mounted on the phone stand with the Raspberry Pi itself for ergonomic and aesthetic purposes.

2.2 Facial Emotion Detection

Due to initial hardware selection, we had originally intended to leverage an existing cloud based API for SSS's computer vision component, with Google's Vision API and AWS being the top candidates. The intention was twofold: to save crucial development time on the computer vision aspect of the process, and to preserve the limited resources allotted to the Raspberry Pi. Although the system was more than capable of running facial recognition-based programs, the nature of the problem statement suggested more rapid processing of data. However, the hourly rate at which the two web services charged proved to be too large an obstacle; due to budget allotments, we would have only had a limited window of time to test the SSS, so the cloud based solution was dropped in favor of physically running the software off of the device itself.

Fig. 1. Google's Vision API in action.

Once the hardware-based solution was selected, we were left with two options: create and train a model of emotion detection from scratch, or utilize an existing one in order to better leverage time and resources elsewhere. Due to the group's relative unfamiliarity with computer vision, paired with an impending testing timeline, it was decided that the software would operate off of an existing emotion detection package. From here, python was established to be the preferred language, in order to offer seamless compatibility with the backend of the interface. After some rudimentary testing with a small selection of emotional detection systems, the open source library titled FER (Facial Expression Recognition) was selected for its compact nature,

perfectly suited to running on a raspberry pi. In short, FER takes in a bounded box of an individual's face, and compares key features to that of the model's database to determine emotion. From there, the emotional components of the result (anger, disgust, fear, happiness, sadness, surprise, and a neutral component), and assigns a confidence value to each. These components could then be converted to those observed in the established psychological model referenced by the project's decision tree, and could be further used to direct the flow of conversation, or act as a check for internally generated estimates.

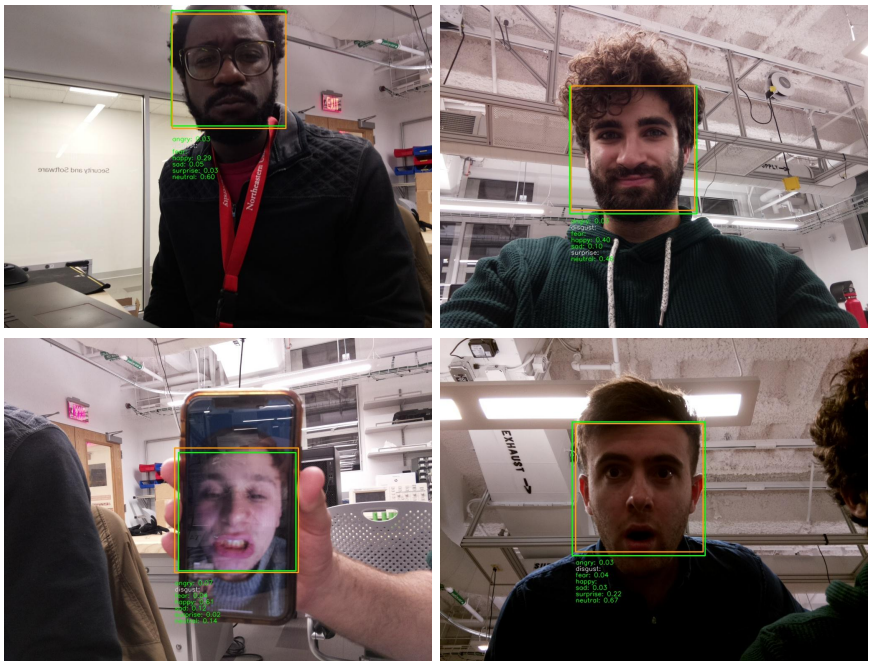

Fig. 2. The software labelling the faces of the team members with the emotions it has detected.

Although it included a rudimentary facial detection system, the strength of FER was in its ability to identify emotions. If used on its own, the library had a difficult time recognizing faces in areas with ever a small degree of foot traffic, and would either ignore a clearly presented face, or return false positives when no person was in frame. In order to amend this, a small program using cv2 (python) was used to identify, and draw bounding rectangles on any identifiable face in frame, and only after doing so, would forward valid images to fer. As pictured above, the SSS's facial recognition operates two times: Once with the handwritten cv2 program (boxes pictured in orange, as seen above) and a second time using FER's systems as a backup measure (boxes in green).

```
    result = detector.detect_emotions(image)
    #Remainder of code only necessary for visual display of info, if the program only needs to know emotions internally, program can stop here
bounding_box = result[0]["box"]
emotions = result[0]["emotions"]

cv2.rectangle(
    image,
    (bounding_box[0], bounding_box[1]),
    (bounding_box[0] + bounding_box[2], bounding_box[1] + bounding_box[3]),
    (0, 155, 255),
    2,
)
```

Fig. 3. CV program returning emotion scores and setting bounding box using FER.

In order to retrieve emotional data for the decision tree, the backend calls this component of the program via function call. Due to hardware limitations, analysis of live video was deemed too intensive, so the program instead takes images at set intervals, until the aforementioned cv2 program can confidently detect at least one person looking directly at the camera. From there, the camera samples multiple images in order to keep an aggregate emotion profile. Only then does FER receive image data and process emotional results, which are returned to the backend. Afterwards, the image data is deleted from the device, in order to preserve users' privacy as much as possible while still processing emotional data.

2.3 Frontend

Overview of Packages

The frontend is built using the React, Node, and Express libraries for JavaScript (JS). The frontend requires 2 ports to be exposed: the first is for the app's GUI, and the second is for an Express-NodeJS API that writes to the Pi's file system and effectively allows the GUI and the backend to communicate. React was chosen as a framework for several reasons. First, there is built-in support for real-time page updates on a certain tick interval. Second, React supports reusable components–which in the SSS's case correspond to several types of question/survey pages–that are not only abstractable but can be built using JSX, an intuitive syntax consisting of JS variable templating and HTML-like markup code. Finally, React is a ubiquitous open-source package whose existing sub-libraries as well as developers' familiarity with it allow for this project to be extended without extraneous effort. Node JS, a backend runtime JS interpreter, allowed us to remain consistent in using JS for the SSS's frontend, including the aspects that required file system access. Express for NodeJS's simple interface for setting up API routes expedited the setup of this communication process.

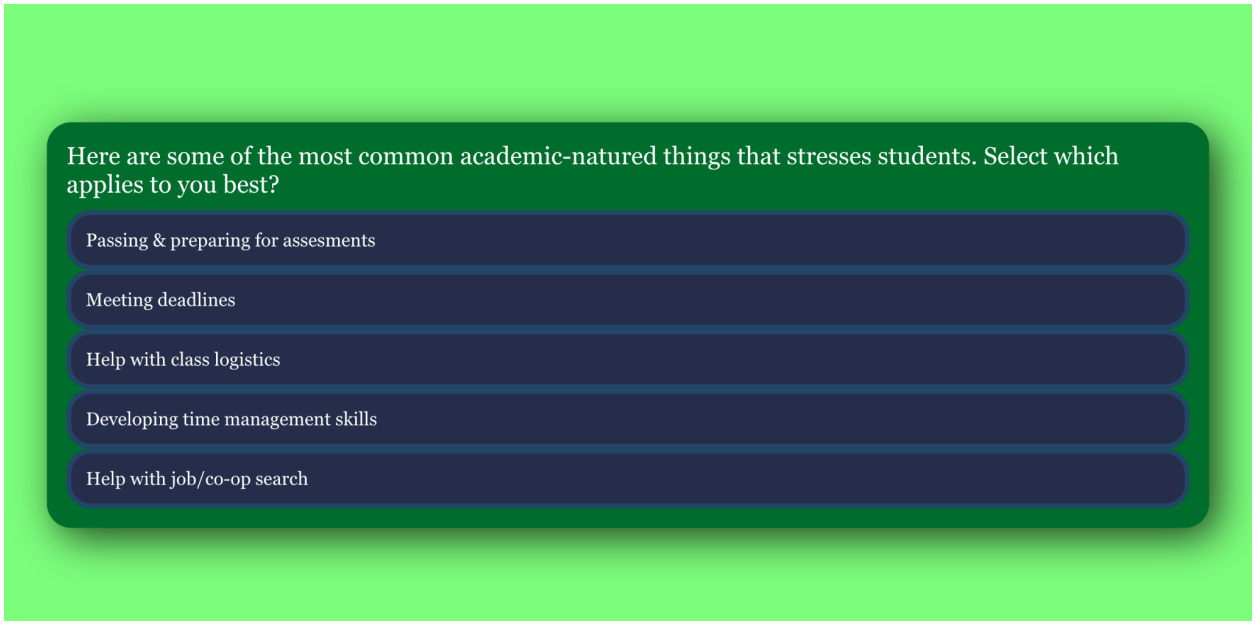

Fig. 4. The frontend asking the user a question and providing possible responses.

Organization

The frontend's functionality resides in a single `frontend/` folder that contains the custom Node packages `fe-main` and `fe-filesys-api/`; the former module contains the main React frontend components and the latter the Express-NodeJS code. Each module utilizes a constants file that exports named constants to the procedural code. Examples of these reusable variables are various directory paths containing frontend-to-backend or backend-to-frontend event transmission files as well as the filesystem api's fully qualified domain name (FQDN).

The fe-main module consists of a highest-level `App` component that initially calls helper `getAndSetData` to HTTP GET the most recent page data and set it to a state variable via React's `useState` hook, and then does so again once the user navigates through the current page. Currently two types of user prompt pages are supported with a corresponding component. The first, the general question page, is usable for both mental health and post-interaction survey questions and HTTP POSTs a user-button-clicked response to the filesystem API. The other page variant is the general solution, which: provides resources, gives the option for that information to be sent to an email specified by a form (then sent to the filesystem API via HTTP-POST), and transitions from psychological content questions to a user experience survey.

```
5   function App() {
6
7     const [currData, setCurrData] = useState({
8       "Prompt type": "Question", // Defaults to a (blank) general question.
9       "Text": "",
10      "Responses": {}
11    })
12    const promptsDict = {
13      "Question": GeneralQuestion,
14      "Solution": GeneralSolution,
15    }
16
17    useEffect(() => {
18      getAndSetData(setCurrData)
19    }, [])
20
21    const CurrPrompt = promptsDict[currData["Prompt type"]]
22    const props = {currData: currData, setCurrData: setCurrData}
23    return (
24      <CurrPrompt {...props} />
25    )
26  }
```

Fig. 5. Frontend code for communicating with the backend.

The fe-filesys-api package hosts an HTTP server, on a route reachable by the main frontend package, that supports HTTP GET and POST requests via callback functions passed into Express's router object. To update the frontend with the correct data: helper `getLatestBeFeTime` is used to scan a specified directory for JSON file basenames with the latest epoch time in format "YYYYMMDDhhmmssuuuuuu" and returns that string (possibly resulting in the same time as before), and then helper `getCurrData` sets that time statefully once there is a new backend-to-frontend transmission. To inform the backend of a user response,

containing or not containing an email address, helper `filewriteQAns` is called to write a JSON file to another specified directory for the backend to process.

```
/**
 * Pass in placeholder latest transmission time to ensure that [1] the latest
 * or [2] any new BE-FE file's basename will be registered as the latest date.
 * The file "0000000000000000.json" should always exist in the
 * BE_FE_PATH_FROM_ROOTas a base case.
 */
var latestBeFeTime = getLatestBeFeTime("0000000000000000", getCurrTime())
// The value of latestBeFeTime preceding an answer post req.
var formerLatestBeFeTime = "" // Should not initially match latestBeFeTime
var currData = getCurrData(latestBeFeTime) // Current page data

router.get("/", function (req, res, next) {
    // Ensure answer is up to date.
    while (latestBeFeTime === formerLatestBeFeTime) {
        latestBeFeTime = getLatestBeFeTime(latestBeFeTime, getCurrTime())
    }

    currData = getCurrData(latestBeFeTime)
    var currData_json = JSON.parse(currData.toString())

    console.log("About to send.", currData_json)

    res.set({
        'Content-Type': 'application/json; charset=utf-8',
    })
    res.send(currData_json)

    console.log(req.headers)
})

router.post("/", function (req, res, next) {
    formerLatestBeFeTime = latestBeFeTime

    // answerUpdated = false
    filewriteQAns(req.body, latestBeFeTime)
    console.log("Received ", req.body)
    // answerUpdated = true
    res.send("Answer received.")
})
```

Fig. 6. Frontend code.

2.4 Backend

Decision Tree

The goal of the decision tree is to recursively ask questions to identify the root cause of the user's negative emotion. All the questions revolve around the 4 core categories of stressors college students face. Once the core stressor of the user's negative emotion is identified, an empathetic solution and general external resources (infographics, articles, videos, etc.) are offered. As an additional feature, a recalculation of the emotion is executed based on the stressors identified.

Content

The contents of the decision tree was sourced from literature surveys that cover common stressors observed in college students [5]-[10]. Using a common classification structure defined by psychology researchers Pariat et al [9], the four core stressors were restrained to Academic, Financial, Social, and Emotional. And using Calaguas [10], Acharya et al [8], the Student Stress Scale [5], and our team members' personal experiences, we populated the possible root issues of each core stressors. Due to lack of time and expertise in its intricacies, we opted out of developing the more general Emotional stressors section. Once the root causes were identified and leading questions were developed, the solution and external resources were collected. To handle an adequate quantity of stressors, the solution and resources provided were constrained to be general, but still insightful. All questions, solutions, and resources were constantly revised to better tailor the content to Northeastern students as feedback was received.

As a method to better approximate the emotion of the user, each specific stressor was mapped to the most probable emotion(s) it might elicit. The emotions spurred from a stressor can be roughly estimated by different arousal and hedonic parameters (Fig. 7).

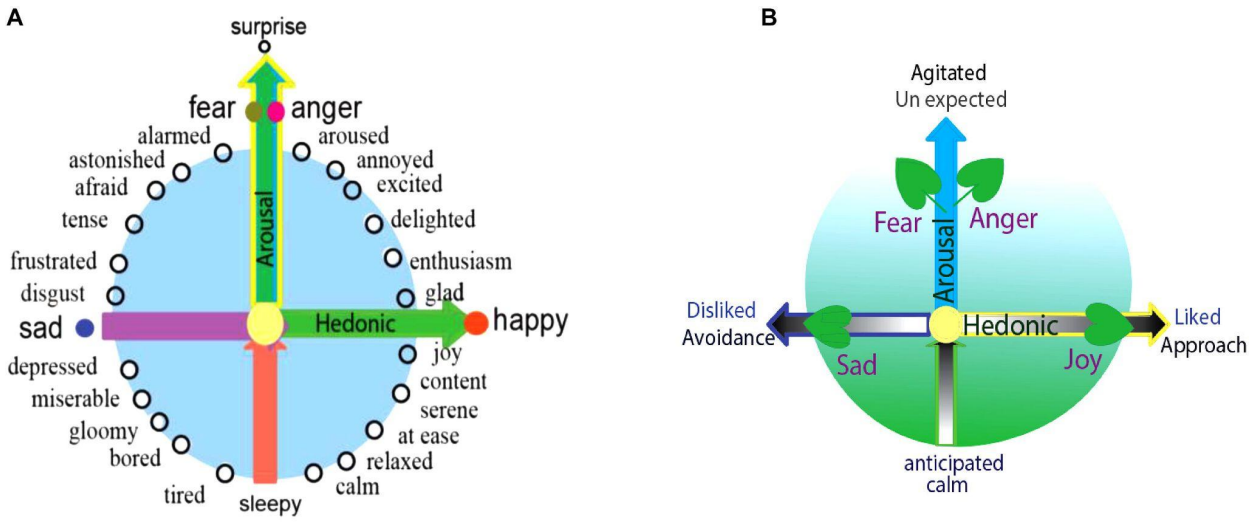

Fig. 7: Core affects and basic emotions [6]

Benefits of this context-based emotion recognition system are that estimation based on context is generally more accurate than just facial emotion detection, and that this setup gives the capability to monitor how closely the decision tree's emotion recognition guess matches the FED approximation and user's actual emotion (based on surveying them).

Organization

All of the questions, solutions, and resources are stored in a Google spreadsheet that contains nine separate subsheets titled: Question Prompts, Response Prompts, Solutions Prompt, Info Listing, Survey Questions, Intro Questions, Academics, Social, and Financial. **Table 1** summarizes the purpose of each subsheet:

Table 1. Descriptions of the sheets in the file storage system.

Sheet Title	Purpose	Used by back end implementation?
Question Prompts	Stores all questions, regardless of their nature (e.g. Academic, Financial). Contains mapping of questions to responses	Yes
Response Prompts	Stores all responses, regardless of their nature (e.g. Academic, Financial). Contains mapping of responses to questions and responses to solutions	Yes
Solutions Prompt	Stores the textual information that is displayed on the solution page of the front end	Yes
Info Listing	Stores the bulleted list of information and general external resources link that is displayed on the solution page of the front end	Yes
Survey Questions	Stores a visual representation of the survey questions to be asked by the front end	No
Intro Questions	Stores a visual representation of the initiating question to be asked by the front end. The intro question varies based on the emotion detected by the computer vision	No
Academics	Stores a visual representation of the Academic-natured questions, solutions, and general resources to be used by back-end and front end.	No
Social	Stores a visual representation of the Social-natured questions, solutions, and general resources to be used	No

	by back-end and front end.	
Financial	Stores a visual representation of the Social-natured questions, solutions, and general resources to be used by back-end and front end.	No

Of the 9 spreadsheets, only Questions Prompts, Response Prompts, Solutions Prompts, and Info Listing are utilized by the backend and frontend to execute the user interaction. These 4 tabs are exported as CSV files which are imported by the backend to generate our decision tree.

Backend-Frontend Communication

The backend receives input from the CV and frontend systems and responds accordingly. When it receives an emotional detection, it searches the database for a question that matches that emotion and pushes that question, along with its possible responses to the frontend. When it receives a user response from the frontend, it searches the database for the correct follow-up question or solution to present. The backend is written in Python3 and sends questions and solutions to the frontend in JSON format.

Table 2. Examples of frontend-backend communication through JSON.

The backend signals the frontend with a question accompanied by the possible responses that the user can give.	{
	"Prompt type": "Question",
	"Responses": {
	"190": "Family",
	"191": "Friends",
	"192": "Roommate",
	"193": "Workplace"
	},
	"Text": "Though challenging, managing interpersonal conflicts is an unavoidable aspect of life. Often the general approach to handling conflicts is the same, but there are subtle nuances based on who it is you have trouble with. Select which type of interpersonal conflict you'd like help with:"
	}

The frontend signals the backend with the response that the user has chosen.	{
	"promptText":"Here are some of the most common social-natured things that stress students. Select which applies to you best?",
	"selectedResponse":
	{
	"182":"Trouble handling interpersonal conflicts"
	}
	}

All possible conversation nodes are given unique IDs and linked to each other. Questions, responses, solutions, and resource links are stored in spreadsheets which are read into the system as comma-separated values. The spreadsheets are designed to be intuitive and simple so a person can customize their system with limited technical or code-related training.

120	Passing & preparing for assesments	13
121	Meeting deadlines	14
122	Help with class logistics	15
123	Developing time management skills	16
124	Help with job/co-op search	17
130	Trouble dealing with test anxiety	1300
131	Unsure how to effectively study	1310
132	Consistently doing poorly on assesments	1320
133	Struggling to understand class material	1330
140	Feeling burnout	1400
141	Struggling with time management	16
150	Unsure about important dates of the semester	1500
151	How to effectively prepare for class registration	1510

Fig. 7. Answers that users may give to questions.

12	Here are some of the most common academic-natured things that stresses students. Select which applies to you best?	120	121	122	123	124
13	Test anxiety, ineffective study method, or course difficulty are some of the common reasons you may find yourself struggling to pass or prepare for assesments. Select which apply to you best?	130	131	132	133	
14	A shaky time management skills is usually the primary factor being difficulty meeting deadlines. However, it's just as likely you may be burnout out from your responsibilities. Select which of these options do you believe apply to you best?	140	141			
15	Whether you're just looking to gauge your upcoming semester's load or reviewing a professor, taking the time to prepare and set-up your classes can some make or break your semester. Which of these option best describe your situation with respect with to class logistics?	150	151			
16	Time management is an important skill to develop as soon as possible. Although challenging to start and maintain, it can lead to massive improvements to your life--incldуing managing classes workload. Which of these option best describe your situation with respect with to time management?	160	161			
17	From preparation to interviewing to coping with rejections, job/coop search can be incredibly stressful. Being able to effectively tackle each of those steps can be helpful to manage the stress that comes with trying to secure a job/coop. Which of these options best describe your situation with respect to time management?	170	171	172		

Fig. 8. Questions that the device may ask.

All conversations are stored in a data file that shows each prompt that was reached and the user response that was selected. User's responses to the survey and the CV emotion detection result are also logged. This format is designed to be easily readable but anonymous so that administrators who are running the system can gain an understanding of the emotions and experiences being expressed by their community members without compromising privacy. Data stored in these log files are able to be rapidly extracted and analyzed through Python scripts.

```
Question|6|Click to begin|2|Begin
Question|10|We've noticed you may be feeling sad, is there anything bothering you today?|100|Yes
Question|11|Select the option that best describes what's bothering you?|110|Academic
Question|12|Here are some of the most common academic-natured things that stress students. Select which applies to yo
Question|16|Although challenging to start and maintain, a good time management routine can lead to massive improvemer
Solution|1600|There are many resources to help with time management. However, the abundance of resources can make it
Question|23|Was this bot useful and engaging?|230|Yes
Question|24|If our assumption of your emotion wasn't correct, please select which emotion you are feeling?|246|Happy
Emotion_CV|sad
```

Fig. 9. Storage of a past conversation.

3. Results

In a sample of 24 users, every solution in the decision tree was accessed at least once. Two solutions were selected three times: the first being in response to interpersonal conflicts with friends, and the second in response to being unsure of where to go with friends in Boston. These two solutions were accessed by 25% of the respondents. Social and academic guidance were the two most commonly selected topics by users, making up 79% of the total interactions. A relatively small proportion of users selected financial issues compared to social or academic issues.

Based on the high concentration of users selecting two of the solutions, more specific questions pertaining to these areas may be useful for future iterations. Refactoring the tree may help guide users towards a stronger solution. With the current approach, users are unable to navigate backwards through the tree. This led to some issues during testing when wrong buttons were selected or follow-up questions were not relevant to the participant.

75% of participants found the SSS to be useful and engaging. The 25% who did not find the device useful were dissatisfied due to a lack of a relevant subject for their situation, or because they misclicked and were unable to reselect their desired answer. This feedback helped identify areas that can be refined in future iterations of the device. The most common feedback about the SSS's functionality was to include a button to go back to change answers. People who found the device to be useful and engaging did not provide much feedback for possible improvements for the project.

One major limitation of our results is that our sample size may have been too small or not representative of all college students. Having feedback from a more diverse range of testers would allow us to have a better understanding of which areas college students are struggling with. As our sample was limited to 24 people, most of which were prior associates of the SSS team, our data may not be comparable with the general Northeastern population. After making modifications to the project based on the last round of feedback, we would like to expand the scale of our testing. An overwhelming majority of testers were members of Northeastern's college of engineering. Collecting data from 100+ students from varying backgrounds and colleges would provide us with a better perspective of which problems plague college students and how the decision tree could be refined.

4. Budget

$185 is required to build our prototype from scratch–far below the limit of $700. For our prototype, we only needed $110 because Professor Padir was able to lend us his lab's Raspberry Pi 4 Model B. Future revisions of the SSS would benefit from a proper enclosure. Only the electronics are included in our budget; the SSS team does not have experience with mechanical design, and therefore are unable to budget for the casing.

Table 3. Budget breakdown.

Materials	Price (Per)	Quantity	Total
Pi Camera Module 2	$32	1	$32
Raspberry Pi 7" Touch Screen Display	$70	1	$70
Raspberry Pi 4 Model B	$75	1	$75
Raspberry Pi Power Supply	$8	1	$8

5. Future Work

The computer vision aspect of the project has vast room for improvement. We had limited time to develop, tune, and optimize the algorithm, leading to slow throughput (up to six seconds to process one image) and high error rate in emotional classification. More time would allow for the model to be optimized for its physical environment and hardware constraints. In all honesty however, the entire computer vision aspect of the project would be re-written from scratch. Between the aforementioned hardware limitations, and the relatively low confidence values exhibited by the model in testing, it would be optimal to use a CV interface fine-tuned to the specifications of the SSS.

The subject matter/content of the project is designed for growth. While the prototype was built for a university environment, future iterations could be optimized for other stressful environments, such as hospitals and warehouses.

Additionally, the project's ease of use could be expanded with more advanced user input techniques. Natural language processing could be used to accept a wider range of user inputs, allowing each user's issue to be narrowed down quickly as they provided more specific answers. There is also the potential to eliminate the need for physical inputs through the use of a microphone plus speech recognition model.

For commercial production, the product would benefit from aesthetic changes. Early in the project, the idea was floated of presenting the device as a friendly character who would be eager to help users with their issues and express empathy toward them. This idea was tabled due to time constraints, but could contribute to the user experience while making the product appear less daunting.

6. References

[1] S. Richardson, J. Shaffer, L. Falzon, D. Krupka, K. Davidson and D. Edmondson, "Meta-Analysis of Perceived Stress and Its Association With Incident Coronary Heart Disease", The American Journal of Cardiology, vol. 110, no. 12, pp. 1711-1716, 2012.

[2] National Institutes of Health, “5 things you should know about stress,” National Institute of Mental Health, 23-Aug-2021. [Online]. Available: https://www.nimh.nih.gov/health/publications/stress. [Accessed: 23-Aug-2021].

[3] S. Vogel and L. Schwabe, "Learning and memory under stress: implications for the classroom", npj Science of Learning, vol. 1, no. 1, 2016.

[4] National College Health Assessment, American College Health Association, Silver Spring, Maryland, rep., 2018.

[5]“Student Stress Scale.pdf.” Accessed: Dec. 16, 2021. [Online]. Available: https://www.baylor.edu/content/services/document.php/183433.p

[6]S. Gu, F. Wang, N. P. Patel, J. A. Bourgeois, and J. H. Huang, “A Model for Basic Emotions Using Observations of Behavior in Drosophila,” Frontiers in Psychology, vol. 10, p. 781, 2019, doi: 10.3389/fpsyg.2019.00781.

[7]K. J. Reddy, K. R. Menon, and A. Thattil, “Academic Stress and its Sources Among University Students,” Biomedical and Pharmacology Journal, vol. 11, no. 1, pp. 531–537, Mar. 2018.

[8]L. Acharya, L. Jin, and W. Collins, “College life is stressful today – Emerging stressors and depressive symptoms in college students,” Journal of American College Health, vol. 66, no. 7, pp. 655–664, Oct. 2018, doi: 10.1080/07448481.2018.1451869.

[9]Ms. L. Pariat, M. A. Rynjah, M. Joplin, and M. G. Kharjana, “Stress Levels of College Students: Interrelationship between Stressors and Coping Strategies,” IOSRJHSS, vol. 19, no. 8, pp. 40–45, 2014, doi: 10.9790/0837-19834046.

[10]G. M. Calaguas, “Survey of college academic stressors: Development of a new measure,” G. M., p. 17, 2012.