We Care!

A Proactive Mental Health Monitoring Website

Budati Yuva Dhatri

200050024

Butti Vaishnavi

200050026

Chilukuri Mani Praneeth

200050028

Desai Divyeswar Reddy

200050033

CONTENTS

CONTENTS ii

1 OVERVIEW 1

2 USERS 2

3 DATA TO BE STORED 4

4 APPLICATION STRUCTURE 9

5 SECURITY ASPECTS 17

6 PLAN FOR GETTING TEST DATA 18

7 FUTURE WORK 20

1 OVERVIEW

In recent years, the suicide rates are very high among students. One of the main reasons for this is their mental health issues caused by external environment i.e., Trauma, Depression. Few of the symptoms of clinical depression are low enthusiasm, fatigue, lack of sound sleep, concentration, frustration, irritability, suicidal tendency. There are many factors that might push students into clinical depression, the following being few of them: Adjustment issues (Far from home, climate difference, language barrier in study medium), No support system - in event of sickness or injury, Academic pressure (high expectations and excellence standards), Financial, Ragging and other harassment

Few of the symptoms can be recognised earlier and can be addressed which might help the individual. So we propose our solution, which is a ”Proactive Health Monitoring Website”. It is a monitoring system that tracks an individual's academic and non-academic activities, and takes feedback from various people in their social sphere, such as friends, batchmates, neighbors, and mentors. Based on the feedback provided, the system calculates an engagement level, which serves as an indicator of the individual's level of involvement and motivation.

If the engagement level drops below a certain threshold, the system sends a notification to the appropriate authorities, such as parents or guardians, to alert them of the situation. This system could potentially be useful in helping individuals stay on track and ensuring they receive necessary support and intervention if they are struggling or disengaged.

However, it is important to consider the potential privacy concerns that may arise from such a system, particularly if sensitive information about an individual's academic or personal life is shared with multiple parties. Additionally, it is important to ensure that the feedback provided is accurate and unbiased, and that the system does not create undue pressure or stress on the individual being monitored. Overall, careful consideration and implementation of such a monitoring system would be necessary to ensure its effectiveness and ethical use.

2 USERS

The roles in our system are as follows:

- Individual being monitored
- Associated People
- Parents or Guardians and Authorities
- Instructors

INDIVIDUAL BEING MONITORED

This user will be the focus of the monitoring system. Their use cases include:

- Logging academic and non-academic activities
- Viewing their engagement level and feedback from associated people
- Editing their profile information
- View their timetable and upcoming deadlines

ASSOCIATED PEOPLE

This user type will provide feedback and ratings on the individual's academic and non-academic activities. Their use cases include:

- Providing feedback and ratings on the individual's academic and non-academic activities
- Viewing the individual's engagement level

PARENTS OR GUARDIANS AND AUTHORITIES

This user type will receive notifications if the individual’s engagement level drops below a certain threshold. Their use cases include:

- Receive mails regarding the individual’s engagement level

INSTRUCTOR

This user type represents the instructors who run courses

- Uploading Attendance CSV of the courses they offer
- View their timetable for the day
- View upcoming deadlines for each course they are offering
- Add deadlines for each course they are offering

3 DATA TO BE STORED

The following are the entities, their corresponding attributes and the data they are storing:

STUDENT

- roll_num: Roll Number of the Student
- name: Name
- dept_name: Department Name
- eng_level: Engagement Level
- hostel: Hostel Number
- room: Room Number
- ec_eng_level: Extra-Curricular Engagement Level

```
CREATE TABLE student (
    roll_num integer,
    name varchar(50),
    dept_name varchar(50),
    eng_level numeric,
    hostel integer,
    room integer,
    ec_eng_level numeric,
    PRIMARY KEY(roll_num)
);
```

FRIENDS

- id1: Roll Number of Concerned Student
- id2: Roll Number of the Friend of this Concerned Student

```
CREATE TABLE friends (
    id1 integer references student(roll_num),
    id2 integer references student(roll_num),
    PRIMARY KEY(id1, id2)
);
```

RECEIVER

This is the table containing data about the receiver of emails if the student's engagement level drops below a threshold.

- stud_id: Roll Number of Student
- name: Name of Receiver
- type: 0 if the receiver is a parent, 1 is the receiver is an authority (faculty advisor)
- mailid: Email ID of the receiver

```
CREATE TABLE receiver (
    stud_id integer references student(roll_num),
    name varchar(30),
    type integer,
    mailid email,
    PRIMARY KEY(stud_id, type)
);
```

TIME_SLOT

- time_slot_id: ID for time slot
- day: Integer indicating the number of day
- start_hr: Starting Hour of the time slot
- start_min: Starting Minute of the time slot

- end_hr: Ending Hour of the time slot
- end_min: Ending Minute of the time slot

```
CREATE TABLE time_slot (
    time_slot_id varchar(4),
    day integer,
    start_hr numeric(2),
    start_min numeric(2),
    end_hr numeric(2),
    end_min numeric(2),
    PRIMARY KEY(time_slot_id, day, start_hr, start_min)
);
```

COURSE

- course_id: ID of Course which is unique for every course
- title: Title of the Course
- dept_name: Name of the Department offering the course
- time_slot_id: ID of the time slot in which course is running

```
CREATE TABLE course (
    course_id integer,
    title varchar(50),
    dept_name varchar(50),
    time_slot_id varchar(4),
    PRIMARY KEY(course_id)
);
```

TAKES

- stud_id: Roll Number of Student
- course_id: ID of Course taken by the Student

```
CREATE TABLE takes (
    stud_id integer references student(roll_num),
    course_id integer references course(course_id),
    PRIMARY KEY(stud_id, course_id)
);
```

INSTRUCTOR

- id: ID of the Instructor which is unique for every instructor
- name: Name of the Instructor
- dept_name: Department Name of the Instructor

```
CREATE TABLE instructor (
    id integer,
    name varchar(50),
    dept_name varchar(50),
    PRIMARY KEY(id)
);
```

TEACHES

- inst_id: ID of the instructor offering the course
- course_id: ID of the course offered by the instructor

```
CREATE TABLE teaches (
    inst_id integer references instructor(id),
    course_id integer references course(course_id),
    PRIMARY KEY(inst_id, course_id)
);
```

DEADLINES

- inst_id: ID of the instructor
- course_id: ID of the course
- start_time: Start Time of the Assignment
- end_time: End Time of the Assignment i.e., the deadline
- name: Name of the Assignment

```
CREATE TABLE deadlines (
    inst_id integer references instructor(id),
    course_id integer references course(course_id),
    start_time timestamp,
    end_time timestamp,
    name varchar(50),
    PRIMARY KEY(inst_id, course_id, name)
);
```

STUDENT_DEADLINES

- student_id: ID of the student
- inst_id: ID of the instructor
- course_id: ID of the course
- name: Name of the Assignment
- done: boolean indicating whether assignment is done

```
    CREATE TABLE student_deadlines (
        student_id integer references student(roll_num),
        inst_id integer,
        course_id integer,
        name varchar(50),
        done boolean,
        PRIMARY KEY(student_id, inst_id, course_id, name),
        FOREIGN KEY (inst_id, course_id, name) REFERENCES deadlines(i
    );
```

AUTH

- user_id: ID of the user
- password: Hashed Password of the user
- type: 0 if he is a student, 1 if he is an instructor

```
    CREATE TABLE auth (
        user_id integer,
        password varchar(50),
        type integer,
        PRIMARY KEY(user_id)
    );
```

4 APPLICATION STRUCTURE

BACKEND

Student Register

- ”post(/register)”
- Parameters Passed:
  - Roll Number
  - Password
  - Name
- Return Value:
  - Returns a JSON corresponding to the INSERT query

Student Login

- ”post(/studentlogin)”
- Parameters Passed:
  - User ID
  - Password
- Return Value:
  - If this is the first time the student is logging in, then it will return the following JSON: {pres: false, result: result}. ”result” is the JSON containing session details.
  - If this isn’t the first time the student is logging in, then it will return the following JSON: {pres: true, result: result}. ”result” is the JSON containing session details.

Instructor Login

- ”post(/instlogin)”
- Parameters Passed:
  - User ID
  - Password
- Return Value:
  - If Login is success, then the JSON {success: ”Logged In”} is returned.
  - Else, the JSON {message: ”Incorrect Credentials!”} is returned.

Student Information Gathering

- ”post(/infogather)”
- If it is the first time a student has logged in, he will be taken to an interface which will gather more of his information
- Parameters Passed:
  - Name
  - Hostel Number
  - Room Number
  - Email ID of Student
  - Department Name
  - Roll Number of Friend 1
  - Roll Number of Friend 2
  - Roll Number of Friend 3
  - Roll Number of Friend 4
  - Roll Number of Friend 5
- Implicit From Session: Roll Number of Student
- Return Value:
  - If there is a session error: {message: ”Session Error!”}

  - If any of the friends aren't yet on the platform: {error: "I am afraid few of your friends haven't yet come aboard on our platform!"}
  - If no errors: {good: "Worked!"}

Parent and Faculty Advisor Details

- "post(/parentdetails)"
- Parameters Passed:
  - Name of Parent
  - Mail ID of Parent
  - Name of Faculty Advisor
  - Mail ID of Faculty Advisor
- Implicit from Session: Roll Number of Student
- Return Value:
  - If there is a session error: {message: "Session Error!"}
  - Else: {success: "Success!"}

Instructor Information

- "get(/instructorinfo)"
- No parameters passed
- Implicit from Session: Instructor ID
- Return Value:
  - If there is a session error: {message: "Session Error!"}
  - Else, a JSON containing the instructor info, the courses he offers, and his timetable for today.

Student Dashboard Display

- "get(/dashdisplay)"
- No parameters passed

- Implicit from Session: Roll Number of Student
- Return Value:
  - If there is a session error: {message: ”Session Error!”}
  - Else, a JSON containing the student details, his course info and upcoming deadline info is sent.

Course Information

- ”post(/courseinfo)”
- Parameter Passed: Course ID
- Return Value: Details of course ”Course ID”

Friends Fetch

- ”get(/friendsfetch)”
- No parameters passed
- Implicit from Session: Roll Number of Student
- Return Value: JSON containing IDs of the students’ friends

Assignments

- ”post(/api/assignments)”
- Parameters Passed:
  - Title of the Assignment
  - Start time and End time of the Assignment
  - Course ID
- Return Value: JSON indicating whether adding assignment is successful

Instructor-Course-Students

- ”post(/api/fetchdata)”
- Parameters Passed:

  - Course ID
  - Instructor ID
- Return Value: JSON containing the details of the students who took course ”Course ID” under instructor ”Instructor ID”

Student-Deadline

- ”post(/api/fetchstud)”
- Parameters Passed: Course ID
- Implicit from session: Roll Number of Student
- Return Value: JSON containing the details of assignments in course ”Course ID” along with the another attribute ”done” which indicates whether the student has done a particular assignment

Assignment-Checkbox

- ”post(/api/handlecheck)”
- Parameters Passed:
  - Course ID
  - Done - Indicating whether the student has checked the checkbox
  - Instructor ID
  - Title of Assignment
- Implicit From Session: Roll Number of Student
- Return Value: JSON indicating success or failure

Edit Instructor Info

- ”post(/editinstinfo)”
- Parameters Passed:
  - Name
  - Department

- Implicit From Session: Instructor ID
- Return Value: JSON indicating success or failure

Update Engagement Level

- ”post(/updateeng)”, ”post(/update_ec_eng)”
- Parameters Passed:
  - New values of engagement level for both
  - Roll Number of Friend for post(/updateeng)
- Implicit from session: Roll Number of Student
- Return Value: JSON indicating success or failure

UI

We are currently planning on supporting a web app only. React is one of the tools we are planning to use for UI.

Key User Interfaces

- Student Sign Up
- LogIn
- Info Gathering
- Parent Details
- Facad Details
- Student Dashboard
- Self-Questionnaire
- Friends Feedback
- Instructor Dashboard

LogIn

This is the first interface that opens up when the website is accessed. The login form takes has two input elements: roll number and password. The user also has to select whether he is a student or an instructor, the default is ”student”. Upon clicking Sign In, the front-end will send a request to the corresponding login API. If the login request is successful, the user will be logged in and taken to corresponding interfaces.

If the user is a student and it is his first time logging in, then clicking on the Sign In button will take him to the Info Gathering interface. If it isn’t his first time logging in, then he will directly be taken to the Student Dashboard. If the user is an instructor, he will be navigated to the Instructor Dashboard interface upon clicking Sign In button.

There is also a Sign Up button present in the right hand side of the page which when clicked will take the user to Student Sign Up page.

Student Sign Up

The form provided has 3 input elements: name, roll number and password. Upon clicking ”Sign Up”, the user will be taken to Login interface if the registration is a success. There is also a ”Sign In” button provided on the left which will take the user to login interface.

Info Gathering

If the student logs in for the first time, he will land at this interface upon logging in. This interface will provide a form collecting Name, Email ID, Department Name, Hostel Number and Roll Number of the student. Upon filling them, the user can click on ”Next” which will take him to another form prompting him for the roll numbers of five of his friends. The Submit button will then submit this information and take the student to Parent Details Interface.

Parent Details and Facad Details

The student will be prompted for details about his parents: Name and Email ID. Upon successfully filling this form and clicking on ”Next” button, he will be taken to the next interface which is Facad Details. He will be prompted for details about his facad: Name and Email ID. Upon clicking submit, he will be navigated to his dashboard.

Student Dashboard

This page displays the following information about the student:

- Personal information: Name, Department, Roll Number, Hostel, Room Number, Engagement Level
- His timetable for the day
- Courses he took

Upon clicking ”Extra-Curriculars” on the navigation bar, the student will be taken to the self-questionnaire and upon clicking ”Friends”, the student will be able to see the list of friends he can give insights on. After filling either of the forms, the backend will check if either of academic engagement level or extra-curricular engagement level of student drops below 30 and if any does, the backend will send a mail to both the parent/guardian and faculty advisor of the student.

The student can click on a course which opens a popup displaying the assignments of that course. The student can then check the checkbox if he finished an assignment.

Feedback - Student, Friends

We have provided forms for each interface which have questions regarding the user. Few of these questions are:

- How often does the individual attend classes
- To what extent does the individual participate in discussions/group activities
- How much does the individual tend to procrastinate regarding the academic activities
- How frequently does the individual study or review course material outside of class time
- How well do you think the individual handles criticism or constructive feedback on their academic work

The answers to these questions are taken on a scale of 1 to 6. Based on the input, we devised an algorithm to calculate how this effects engagement level.

Instructor Dashboard

This is the landing page for successful instructor login. This page displays the following information about the instructor:

- Personal Information: Name, ID, Department
- The Courses he is offering
- His timetable for today

There is also an ”Add Assignment” option which opens a popup form to create a new assignment.

5 SECURITY ASPECTS

SQL INJECTION

SQL injection is a major security threat we need to address. Hackers can inject malicious SQL query via input data and this malicious query when validated will grant access to the hacker to potentially act as database administrator. Concatenating SQL query strings to the input is one of the main loopholes that a hacker can use to manipulate these SQL queries and hence expose data.

To counter this, we used the ”Prepared Statement Approach”, also known as Question Mark Approach. The advantage of using the question mark is, it will match the data comparison only with the particular column. Even if the hackers modify input, the modified input will be compared with only one column. So it does not expose the data.

SESSION VULNERABILITIES

Session vulnerabilities arise in multiple situations but the ones we addressed in our website are:

- **Session Reply Attack** : User session being left active even after the user logs out. In this case, the attacker can acquire the session token and use the same to log into the webpage without user credentials.
- **Insufficient Session Expiration** : This happens when a web site permits an attacker to reuse old session credentials for authorization. This can can happen if the session doesn’t expire when the user forgets to logout.

6 PLAN FOR GETTING TEST DATA

Since we weren't able to find data which serves our purpose, we planned on creating it on our own. For testing, we created a database about the same size of our small University database. There are a total of 10 tables in our database.

STUDENT

As there are a few constraints that the student register option of our website demands, we won't be able to start from scratch and build the student table just through our website. So, the data of students is a combination of both pre-fed data and registered data. The pre-fed data is manually uploaded by us to the database. The registered data is the data of new students who register through the website.

FRIENDS

Data to this table is added through the "Info Gathering" interface of website.

COURSE

This table is similar to the Course table in university database. So we just picked few of the tuples from that database.

TAKES

Once the student is added to the student database, his course data is added to the takes table(manual entry by us). We took few tuples from the university database as it matches our requirements perfectly and also reduces the mental load to think of new names and IDs for courses :)

INSTRUCTOR, TEACHES

We extracted this data from the university database and is manually entered into database by us and not through the website.

RECEIVER

When a student is logging in for the first time, he will be prompted to provide parent and facad details. These details are stored in this table. We haven’t added any pre-fed data.

MENTORS

Similar to Receiver, this data is prompted from the student and will be entered into the database through the website. No pre-fed data.

DEADLINE

Instructor will be adding assignments through the website which are then entried into this table. No pre-fed data.

AUTH

For students, data is added into this table through register option of the website. Coming to instructors, we haven’t provided a register option for them and hence the authorization data of instructor is manually added by us

7 FUTURE WORK

Attendance

Instructor can upload CSV file of attendance and we can calculate the attendance of the student in the current week using this. We can then change the engagement level accordingly.

Appointments

We can add an Appointment Feature where the student can book appointment to doctors of Student Wellness Centre. When the student's engagement level drops below a threshold, the doctor will also be notified.