LAB REPORT

Submitted by

Mayank Kumar [RA2011003010600]

Under the Guidance of

Ms. G.K. Sandhia

Assistant Professor, Computer Science and Engineering

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in CORE SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

JUNE 2022

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this lab report titled **Software Engineering and Project Management** is the bonafide work done by Mayank Kumar (RA2011003010600) who carried out the lab exercises under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE

Ms. G.K. Sandhia

SEPM – Course Faculty
Assistant Professor

Department of CSE

ABSTRACT

We live in a world which tends to have an overwhelmingly fast pace, finding solace and a relaxing pace in midst of that, seems almost like a necessity.

“Flow” is an app that aims to provide just that. With an easy-going interface, and flexible approach which makes it more personalized; Flow is flexible, efficient and comforting.

At its core, Flow is a music therapy app that lets users choose between three modes of “Relax”, “Focus” and “Sleep”.

With this app, we aim to provide everyone a platform to create a safe space of growth, healing and focusing on betterment of oneself.

TABLE OF CONTENTS

LIST OF FIGURES

Project Description

THE PROJECT

In bullet points, describe the problem this project aims to solve or the opportunity it aims to develop.

● Often do people feel the need to hide their problems from people close to them, so as to not bother them.

● Thus an app, that connects people to random strangers with anonymous ID's and let them connect once

and only once based on their answers to pre-defined questions.

● This app also has options to create playlists, send audible e-books, and white sound playlists and share them.

● It is a well-being centred social app.

THE HISTORY

In bullet points, describe the current situation.

● In the current situation, people often face issues with opening up and approaching people close to them so as to not cause them issues.

● This often causes people to pent up issues and emotions that splutter out at wrong times, in wrong situations.

● A place to vent these emotions, ANONYMOUSLY, at that, can be very helpful.

● An added key benefit is the fact that the people connected once aren't EVER connected again.

LIMITATIONS

List what could prevent the success of the project, such as the need for expensive equipment, bad weather, lack of special training, etc.

● Time-consuming to make

● Two people who know each other personally might match thus dissolving the main mission of the app.

APPROACH

List of what is needed to complete the project.

● API Tracking

● Real-Time analysis

● JSON

BENEFITS

In bullet points, list the benefits that this project will bring.

- Peace
- Mental Well Being

Stakeholders And Process Models

1.Executive Summary

1.1 Project Methodology:

Agile Methodology is a people-focused, results-focused approach to software development that respects our rapidly changing world.

It's centered around adaptive planning, self-organization, and short delivery times.

It's flexible, fast, and aims for continuous improvements in quality, to put in simple terms, Agile helps teams in delivering value to customers quickly and effortlessly.

Thus, developing such a software management project, Agile methodology brings out the most effective growth out of it.

1.2 Stakeholder Identification:

Internal stakeholders:

The internal stakeholders include the team members, managers, executives who are all internally related.

Project Role	Responsibilities	Team members assigned to
Project Manager	(1) Plan and develop the overall project idea. (2) Create and lead our team in all aspects. (3) Ensure stakeholder satisfaction.	Shreya Khera
Technical Lead	(1) Collaborating with the team to identify and fix technical problems. (2) Analyzing the user's needs and then finding the applications to serve them.	Rushil Kothari

Business Analyst	(1) Budgeting and forecasting. (2) Variance analysis (3) Defining business reqts. and reporting them back to stakeholders.	Mayank Kumar
Developer	(1) Researching, designing, implementing, and managing software. (2) Writing and implementing efficient code.	Mayank Kumar Shreya Khera Rushil Kothari
Tester	(1) Develop and execute tests. (2) Troubleshoot issues. (3) Document and code bugs.	Mayank Kumar Shreya Khera Rushil Kothari

External stakeholders:

Project Role	Responsibilities	Team members assigned to
Suppliers	Accountability for quality issues. Ethical and regulatory compliance.	Shreya Khera
Society	Fulfilling civic duties.	Rushil Kothari
Government	Provide funds as venture capital. Encourage banks and other financial institutions to provide venture capital.	Mayank Kumar
Creditors	Extends credits by giving another entity permission to borrow money intended to be repaid in the future.	Shreya Khera Rushil Kothari Mayank Kumar
Customers	Be an ethical consumer. Share experience and respect the environment.	Anyone

2.Stakeholder Management

2.1 Interest and Influence matrix

Interest	Influence
High	High
Low	Low
Low	High
High	Low

2.2 STAKEHOLDER INTEREST, INFLUENCE, PRIORITY IDENTIFICATION

Stakeholder	Responsibility/Interests	Influence	Estimated Priority
Owner	Achieve targets, increase sales margin	High	1
Sponsor	Provides new markets to expand ventures. Negotiate funding for project. Reviews changes to project environments.	Med	3
Team members	Demand incentives. Retain and upgrade skills. New product excitement.	High	2
Project Manager	Lead the team in every aspect. Accountable for entire project scope, team, success & failure.	High	2
Investors	Promoter of the investment. Provides necessary financial resources.	Low	5
Resource Manager	Resourcing planning and allocation. Ensuring adequate resource according to project needs and budget.	Med	4
Suppliers	Ensuring feasible and realistic in every aspect. Managing divergence from budgeted cost.	Med	6
End Users	Provides feedback	Low	7

Identifying Requirements

User Requirements:

The user will definitely require an active E-mail ID. They will also be required to choose a unique username, i.e., it cannot be repeated by anyone, this username should also make their identity anonymous. To use the playlist functionality, the user will also have to have an active Spotify ID with the same email.

System Requirements:

- iOS/Android system
- Active Internet Connection

Functional Requirements:

- Login Page

- “Personality Check” Page
- “Connect to Random” option
- “Choose to make playlist” option
- “Audio-books” option
- “Mental Health Podcasts” option

Non-Functional Requirements:

- Enough people on the app to never connect to once connected person again.
- Lack of Spotify ID, so unavailability in using “Playlist” functionality.

Project Plan and Effort

Project Management Plan

Focus Area	Details
Integration Management	• Governance Framework - Guidance system composed of standard management practices within governance framework to suit the organization. • Project Team Structure - The project team includes the team lead and its team who work together to achieve a common goal. • Project Closure - Project closure is the critical last phase of the project development.
Scope Management	• Scope Statement - Project scope is a target which we wish to achieve before we run out of our project management plan. • Requirement Management (Gathering, Control, Assumption, Constraint Stakeholder) - Requirement management is the process of ensuring that the demands or requirements of internal and external stakeholders to finish the project is fulfilled. • Define deliverable - Deliverables are the services or the product that we need to deliver in between and at the end of the project.

Schedule Management	• Define Milestones - Each step in the life cycle of project management is a milestone. Such as: Feasibility study, Requirement gathering, planning, designing, developing, testing, maintenance etc. • Schedule Control - Schedule control is to manage time in a project. This is done to monitor that you are proceeding as planned.
Cost Management	• Estimate Effort - Effort estimation is when a product project development team estimates how much effort is required to build the project. • Budget Control - Budget control is a process of planning the budgetary for the future, it is like predicting how much budget will be required to complete the project. Budget control lets us stay within our budget margin.
Quality Management	• Quality Assurance: Quality assurance will be managed including governance, roles and responsibilities, tools and techniques and reporting
	• Quality Control: Specify the mechanisms to be used to measure and control the quality of the work products
Stakeholder	• Internal stakeholders - 1. Project manager 2. Technical lead 3. Developer 4. Business analyst 5. tester • External stakeholders - 1. Customer 2. Supplier
Risk Management	Risks identified - 1. Corrupt code 2. Server breakdown 3. Lack of requirements 4. Lack of commitment

1. Estimation

1.1. Effort and Cost Estimation

Activity Description	Sub-Task	Sub-Task Description	Effort (in hours)	Cost in INR
Design the user interface	User Requirement Confirmation	Confirm the user requirements (acceptance criteria)	5	2500
	Designing	Designing the UI	40	20000
	Review	Reviewing the UI for final adjustments	10	5000
Developing Front End	Develop Basic skeleton	Go through the designed UI and code the basic skeleton of the platform	7	3500
	Add UI Elements	Add more complex UI elements and make website responsive and dynamic	50	25000
Developing Back End	Make UI	Make the UI	30	15000
	interactive	elements interactive		
	Develop core functionalities	Implement core functionalities of the platform	110	55000
	Integrate with DBMS	Integrate the platform with a DBMS system for data storage	10	5000
	Debug	Extensively test and debug the platform	20	10000

Deployment	Test run	Deploy it in openbeta for user and developers to test out the platform in real world scenarios	10	5000
	Deploy publicly	Make final changes as per the test run results and deploy it stably for public	10	5000

Effort (hr)	Cost (INR)
1	500

1.2. Infrastructure/Resource Cost [CapEx]

Infrastructure Requirement	Qty	Cost per qty	Cost per item
Laptop	3	50000	150000
Smartphone	3	20000	60000
Network Connection	3	2000	6000

1.3. Maintenance and Support Cost [OpEx]

Category	Details	Qty	Cost per qty per annum	Cost per item
People	Network, System, Middleware and DB admin Developer, Support Consultant	3	2,000,000	6,000,000
License	Operating System Database Middleware IDE	10	10000	100,000
Infrastructures	Server, Storage and Network	20	20000	400,000

2.Project Team Formation

2.1. Identification Team members

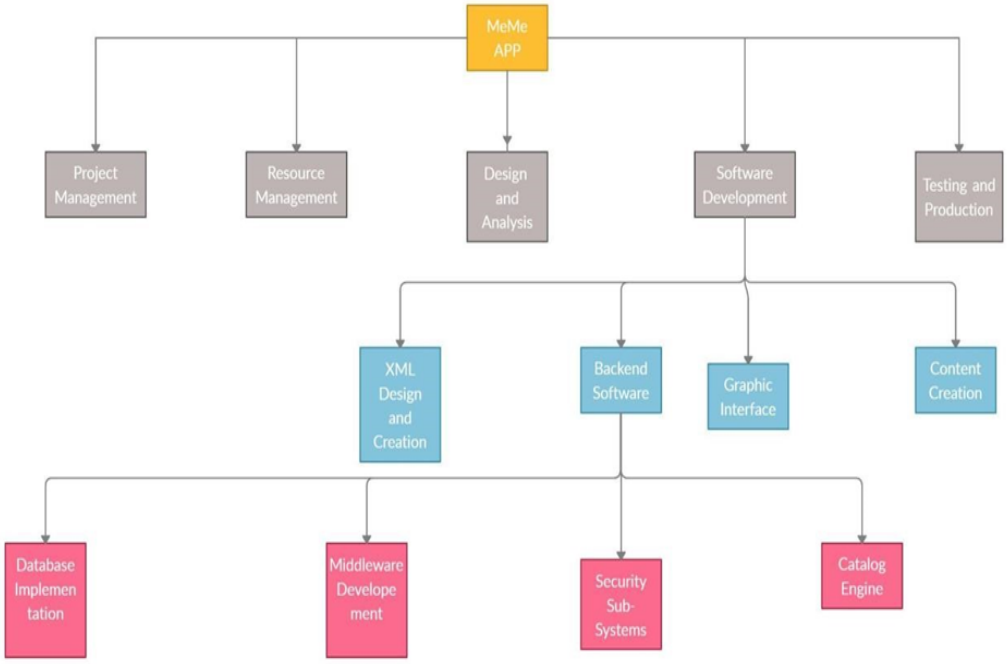
Name	Role	Responsibilities
Mayank Kumar Shreya Khera Rushil Kothari	Key Business User (Product Owner)	Provide clear business and user requirements
Shreya Khera	Project Manager	Manage the project
Mayank Kumar	Business Analyst	Discuss and Document Requirements
Rushil Kothari	Technical Lead	Design the end-to-end architecture
Mayank Kumar	UX Designer	Design the user experience
Rushil Kothari Shreya Khera	Frontend Developer	Develop user interface
Shreya Khera Rushil Kothari	Backend Developer	Design, Develop and Unit Test Services/API/DB
Mayank Kumar Shreya Khera Rushil Kothari	Cloud Architect and Operations	Design the cost-effective, highly available and scalable architecture and Provision required Services
Mayank Kumar Shreya Khera Rushil Kothari	Tester	Define Test Cases and Perform Testing

2.2. Responsibility Assignment Matrix

RACI Matrix	**Team Members**			
Activity	**Mayank Kumar (BA)**	**Rushil Kothari (Developer)**	**Shreya Khera (Project Manager)**	**Key Business User**
User Requirement Documentation	A	C/I	I	R
Project Management	R	C/I	A	I
Designing	I	A	R	C/I
Development	I	A	R	I
Deployment	C/I	A	R	I

A	Accountable
R	Responsible
C	Consult
I	Inform

Work Breakdown Structure



1. Gather Requirements
    - 1.1 Technical specifications
    - 1.2 User requirements
    - 1.3 Reporting requirements
2. Establish design
    - 2.1 Design elements
    - 2.2 Overall layout
    - 2.3 Content elements
3. Selest technical framework
    - 3.1 Evaluate options against requirements
    - 3.2 Evaluate cost and time to develop
    - 3.3 Make decisions
4. Implement technical framework
    - 4.1 Build or acquire back end
    - 4.2 Build or acquire front end
    - 4.3 Combine front end and back end
5. Create content
    - 5.1 Create content summary
    - 5.2 Establish content details
    - 5.3 Assign content creation
    - 5.4 Create detailed content
6. Load content
7. Test site
    - 7.1 Navigation
    - 7.2 Interactive element
    - 7.3 Browser compatibility
8. Roll out site

8.1 Establish target date
8.2 Create communication plan
8.3 Make site live

2.0 Requirements Gathering

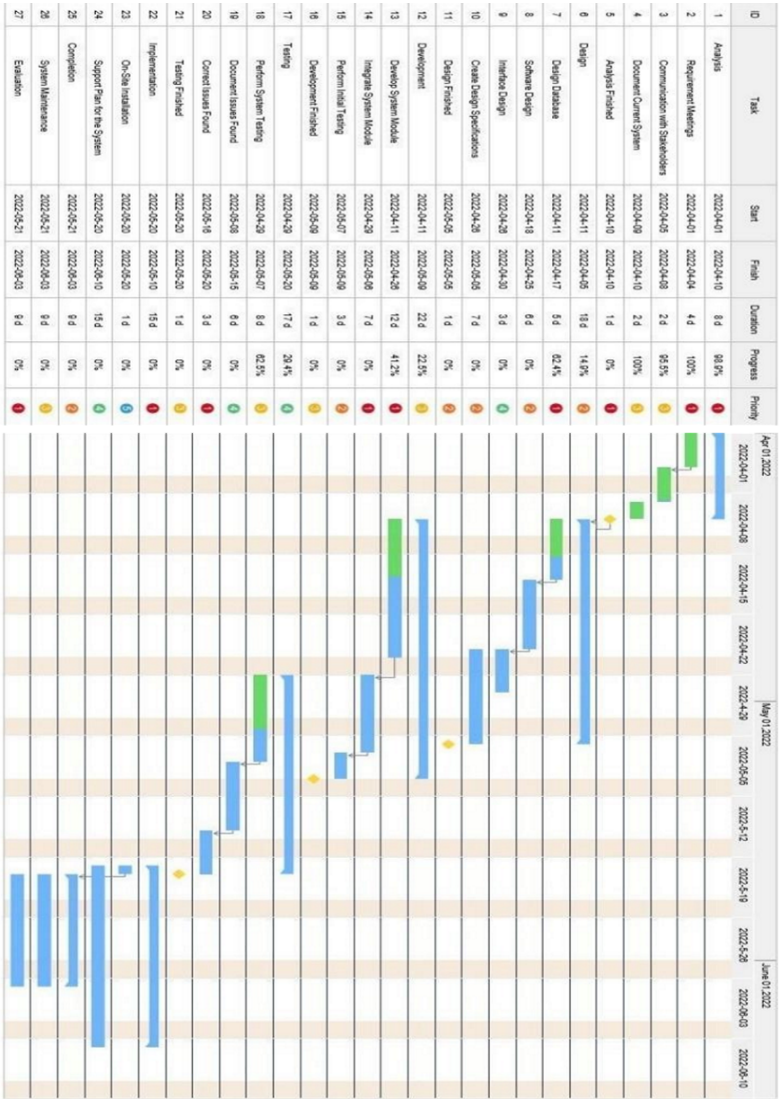
3.0 Analysis & Design

4.0 Site Software Development

- 4.1 HTML Design and Creation
- 4.2 Backend Software
  - 4.2.1 Database Implementation
  - 4.2.2 Middleware Development
  - 4.2.3 Security Subsystems
  - 4.2.4 Catalog Engine
  - 4.2.5 Transaction Processing
- 4.3 Graphics and Interface
- 4.4 Content Creation

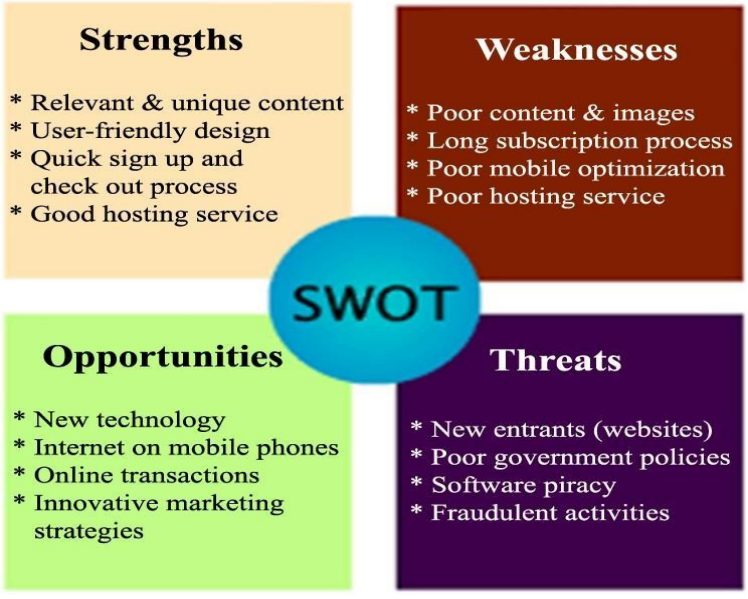
5.0 Testing and Production

TIMELINE – GANTT CHART

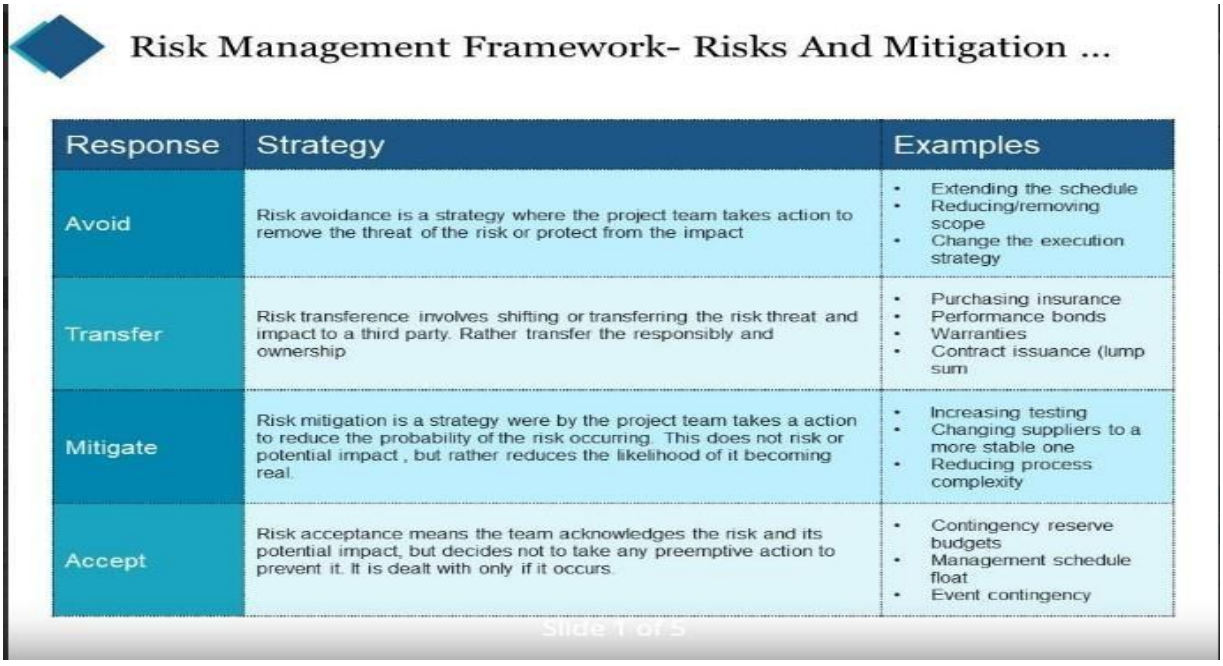


ID	Task	Start	Finish	Duration	Progress	Priority
1	Analysis	2022-04-01	2022-04-10	8 d	98.9%	0
2	Requirement Meetings	2022-04-01	2022-04-04	4 d	100%	0
3	Communication with Stakeholders	2022-04-05	2022-04-08	2 d	95.5%	0
4	Document Current System	2022-04-09	2022-04-10	2 d	100%	0
5	Analysis Finished	2022-04-10	2022-04-10	1 d	0%	0
6	Design	2022-04-11	2022-04-25	10 d	14.0%	0
7	Design Database	2022-04-11	2022-04-17	5 d	62.4%	0
8	Software Design	2022-04-18	2022-04-25	6 d	0%	0
9	Interface Design	2022-04-26	2022-04-30	3 d	0%	0
10	Create Design Specifications	2022-04-26	2022-05-05	7 d	0%	0
11	Design Finished	2022-05-05	2022-05-05	1 d	0%	0
12	Development	2022-04-11	2022-05-09	22 d	22.5%	0
13	Develop System Module	2022-04-11	2022-04-25	12 d	41.2%	0
14	Integrate System Module	2022-04-29	2022-05-06	7 d	0%	0
15	Perform Initial Testing	2022-05-07	2022-05-09	3 d	0%	0
16	Development Finished	2022-05-09	2022-05-09	1 d	0%	0
17	Testing	2022-04-29	2022-05-20	17 d	29.4%	0
18	Perform System Testing	2022-04-29	2022-05-07	8 d	62.5%	0
19	Document Issues Found	2022-05-08	2022-05-15	6 d	0%	0
20	Correct Issues Found	2022-05-16	2022-05-20	3 d	0%	0
21	Testing Finished	2022-05-20	2022-05-20	1 d	0%	0
22	Implementation	2022-05-20	2022-05-10	15 d	0%	0
23	On-Site Installation	2022-05-20	2022-05-20	1 d	0%	0
24	Support Plan for the System	2022-05-20	2022-06-10	15 d	0%	0
25	Completion	2022-05-21	2022-06-03	9 d	0%	0
26	System Maintenance	2022-05-21	2022-06-03	9 d	0%	0
27	Evaluation	2022-05-21	2022-06-03	9 d	0%	0

RISK ANALYSIS – SWOT & RMMM

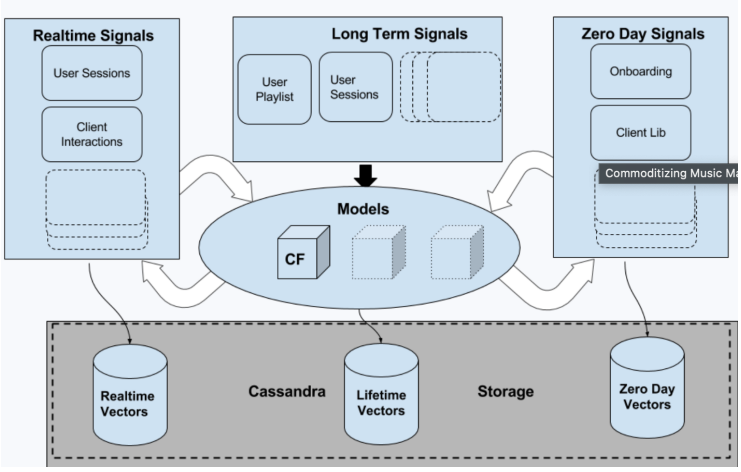


Strengths	Weakness	Opportunities	Threats
● Reliable Vendors ● Marketing and promotional ads ● Price range, discounts and Memberships	● Lack of Manpower ● Competitive market	● More population and higher opportunities to earn ● profitable ● jobs for many jobless	● Expensive ● Lack of long term contracts ● Should follow newly implied government regulations



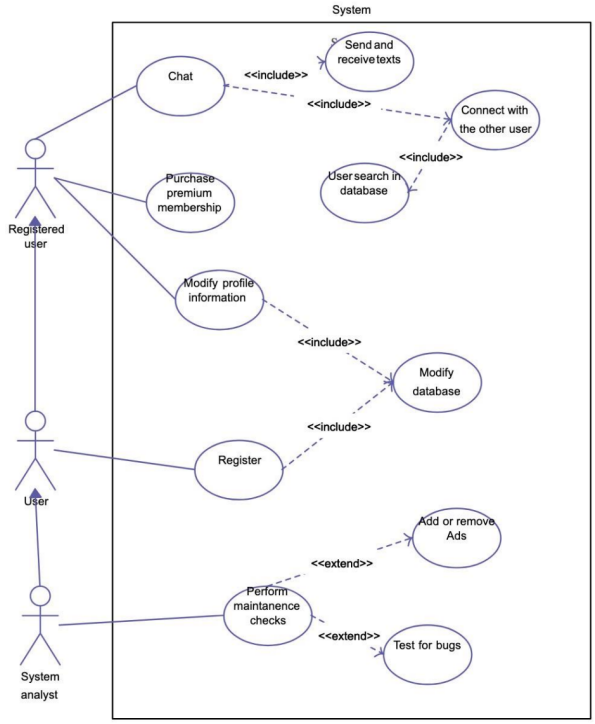
Parameter	Risk	Cause	Control
Human Resource	Serious injuries, death, disability	Work accident	- First aid in accidents - Emergency room - Insurance - Rehabilitation
Asset	The stoppage of the production process	Damage to equipment	- Periodical checking to the equipment - Improvement planning to the damage equipment - New equipment
Information Technology	Stranded at the operational process	Damage to hardware and software	Routine checking
Nature Phenomena	Dysfunction, serious injuries, death, disability	Natural disasters	Insurance

System Architecture



The architecture diagram of our project shows the layout of our project. From the diagram we can infer that the app which is made using Flutter and firebase is integrated using a serverless integrator and is connected to the server. The server has access to the database and it redirects all the incoming and outgoing links to the security module which manages all the data links as it sends the client script to the client PC and receives the client searches via cloud link. The client script ultimately shows the desired data to the client app.

Use Case Diagram



In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

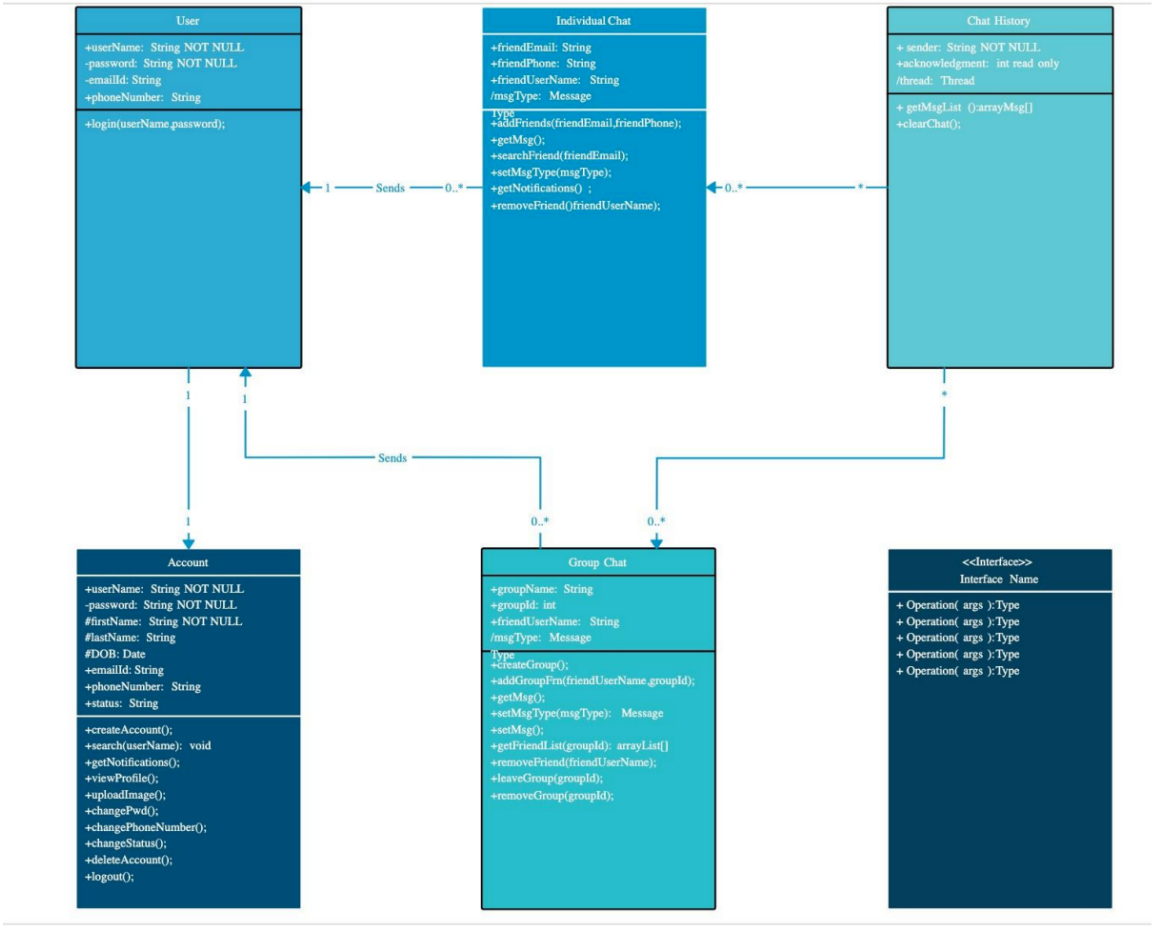
● Scenarios in which your system or application interacts with people, organizations, orexternal systems

● Goals that your system or application helps those entities (known as actors) achieve.

● The scope of your system

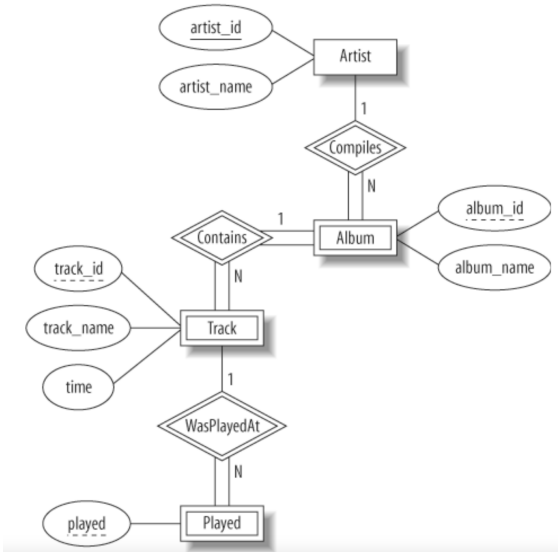
Class Diagram

Class diagram is a static diagram. It represents the static view of an application. The class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

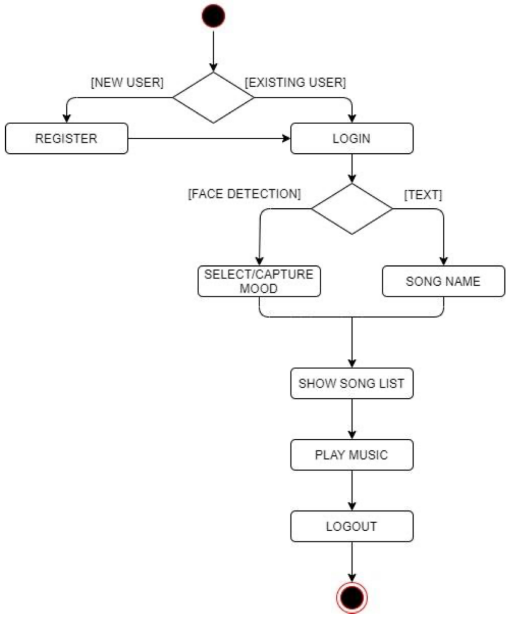


Entity-Relationship Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

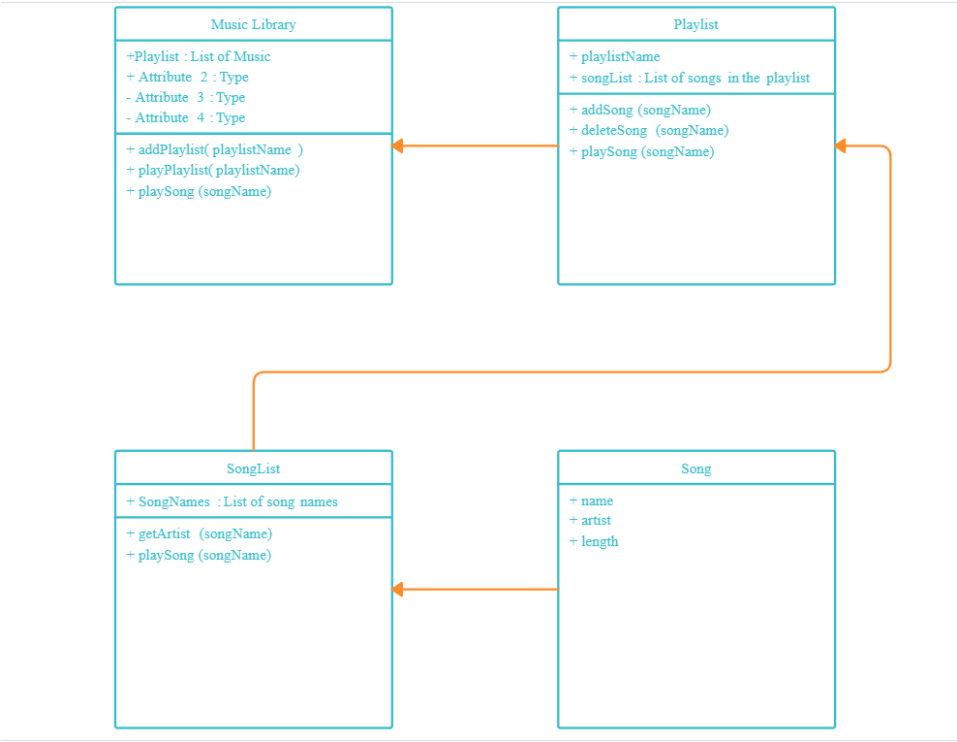


Data Flow Diagram



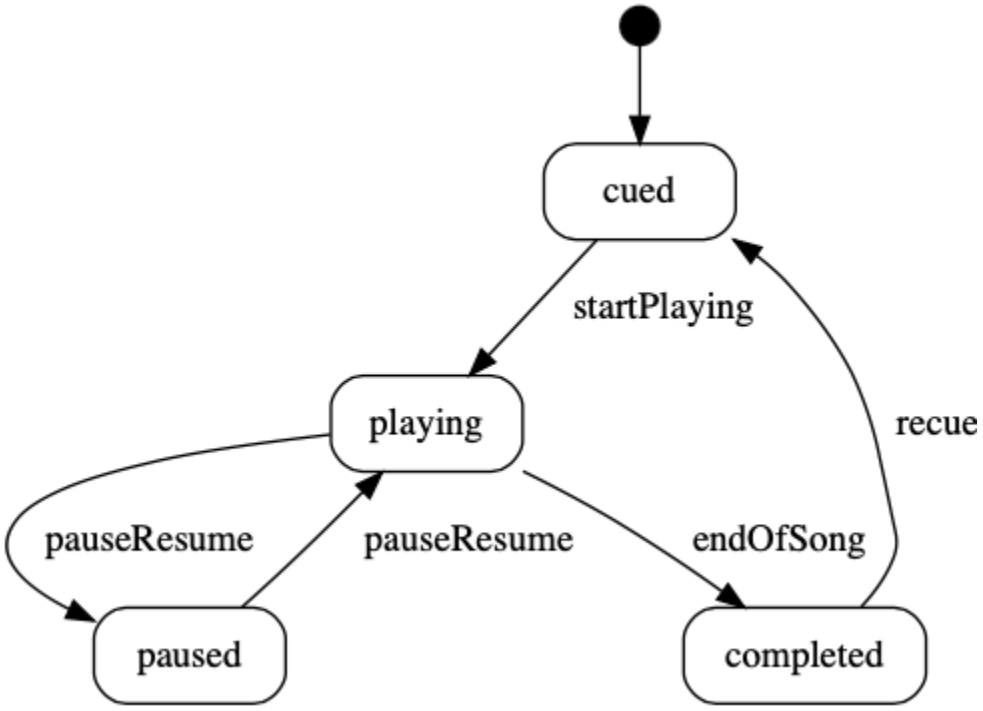
Data flow diagrams (DFDs) reveal relationships among and between the various components in a program or system. DFDs are an important technique for modeling a system's high-level detail by showing how input data is transformed to output results through a sequence of functional transformations.

Collaboration Diagram



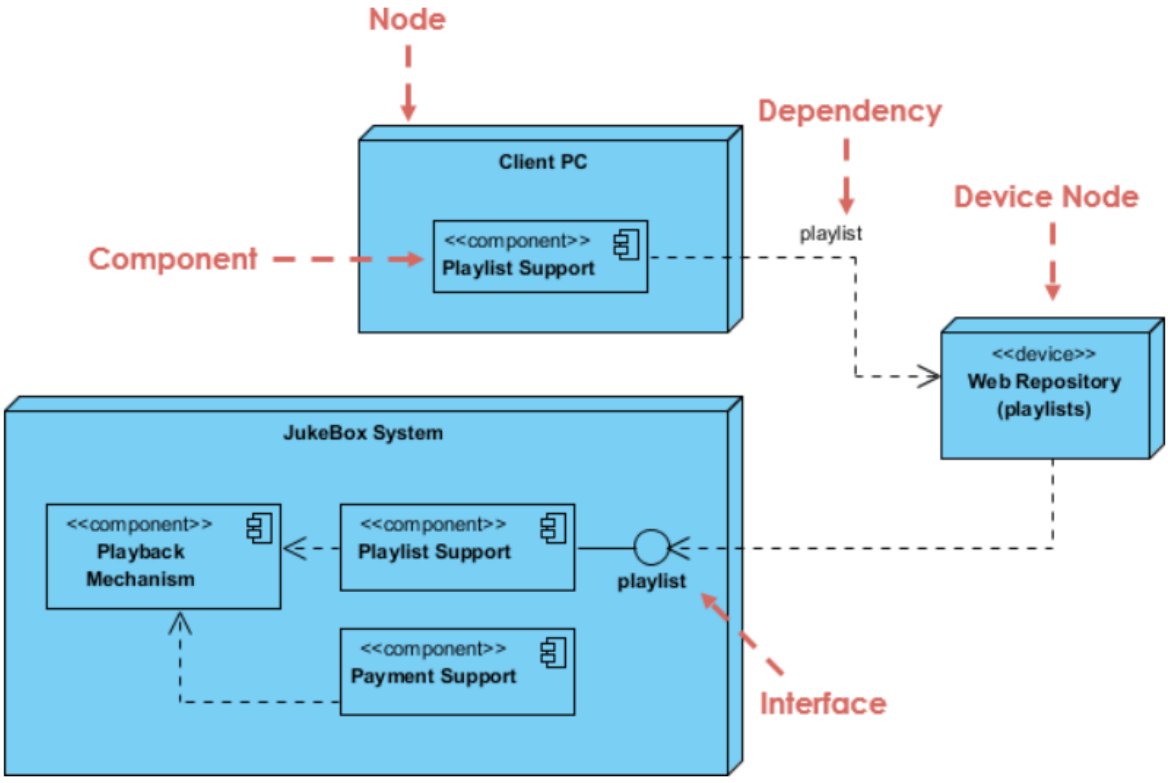
A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

State Diagram



A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

Deployment Diagram



This deployment diagram example shows the deployment topology of a music Jukebox System and the relationship among the nodes: client PC, Web Repository and the Jukebox System. Each of the deployment nodes resides the component(s) which deployed to them.

Test Cases and Reporting

Executive Summary

In this new age of medicine, everything is being heavily digitized. It makes more sense to look up solutions on the internet

instead of actually going through the process of seeing doctors offline.

And it makes more sense in the case of mental health consultation.

People who suffer from mental health issues tend to turn for help to sources that involve less human interaction.

This may be because it takes time to open up to a stranger, that too in person.

As technology advances, more and more apps are being created to

serve patients who are struggling with mental health disorders.

The patients are looking for mental health software, similar to these apps that help them cope with their mental well-being.

However, not all apps end up being the most effective.

Test Plan

Scope of Testing

The testing will cover testing the home page, signup/login page and post page on things like entering a nickname / alias , creation and popping of a chatbot for 2 random strangers to talk and ease their mental health, etc. and testing various non-functional requirements like speed, performance, delay.

Functional: All functional requirements are being taken care of. For ex: - functioning of home page, login/signup, chatbot initiation, etc.

Non-Functional: Almost all non- functional requirements are being taken care of. For ex: - Performance, Speed, Time delay, huge traffic of users.

Types of Testing, Methodology, Tools

Types of Testing, Methodology, Tools

Category	Methodology	Tools Required
Functional Requirements	Manual	Word Template
Non-Functional Requirements	Manual	Word Template

Test Case

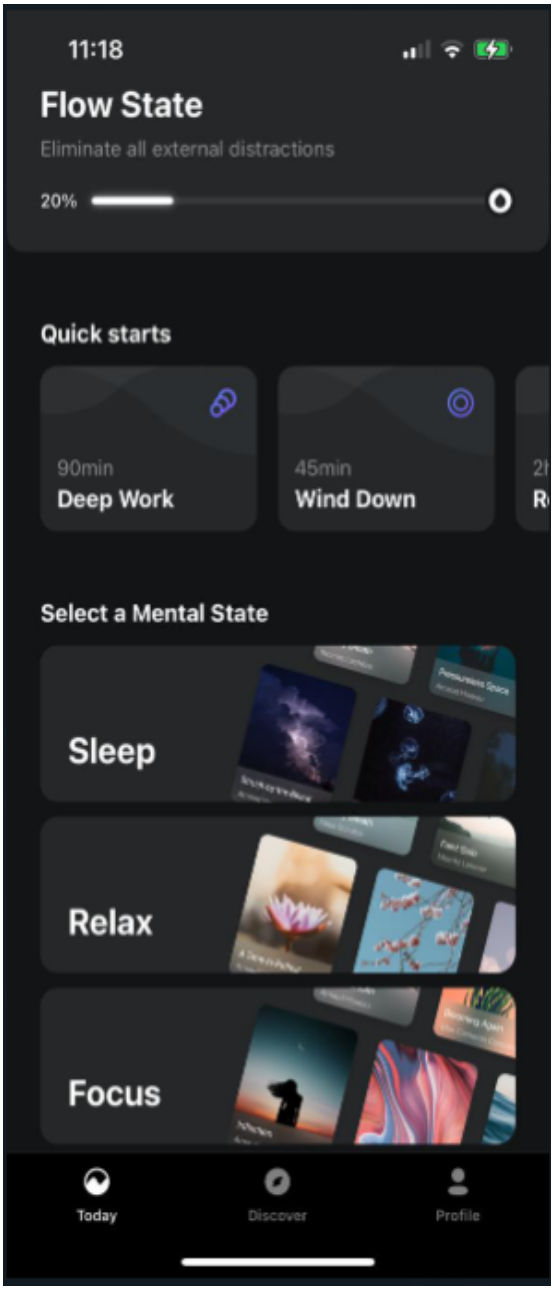
Functional Test Cases

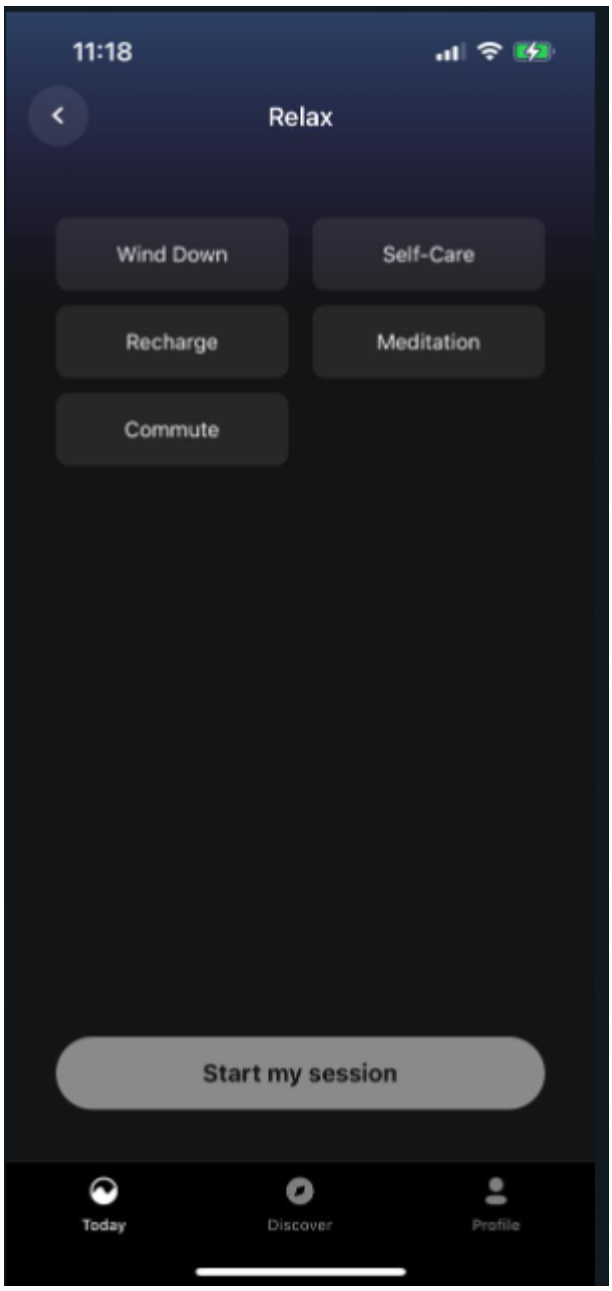
Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
T_001	Check signup functionality	Accepts a nick name or alias for the user to ensure the privacy for the user	1. User clicks on signup 2. Enter the valid details 3. Click submit button	User should be taken to the next page where they can create new post	User goes to next page where they can create new post	Pass	success
T_002	Check chatbot functionality	Check if a chatbot is popped once the user provides their nickname or alias	1. User provides with his/her nick name or an alias 2. Enter nick name 3. Click submit button	User will be shown error upon not entering any ni	User gets shown wrong password after entering invalid password	Pass	success
T_003	Connect with a random user and chat while maintaining anonymity	Accept the user's details and initiates the chat	1.User clicks on the chat bot 2. User initiates the chat with the random person 3. Click submit button to submit their chats	User will be shown a tick mark which indicates the success of message being sent	User sends the message and the other person too reads it successfully	Pass	success

Non-Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
T_001	Multiple users interacting at the same time and leads to a lot of traffic	Accepts a valid user and allows it to initiate the chat with the other user.	1.Users will click on chatbot initiation button. 2.Fill necessary details like aliases or nickname 3.Press Submit button to initiate the chat between the 2 strangers	All users will go to other page where all the posts are displayed	All users go to other page where all the posts related to mental health awareness are posted. Additionally playlists related to mental health awareness are also provided with.	Pass	Success
T_002	Users visit the home page	Enter the website link to visit the page	1.Users will type the website link. 2.User press enter.	Users will go to the root directory with a neat and beautiful user interface	Users goes to root directory with a neat and beautiful user interface.	Pass	Success

Sample Frontend design





CONCLUSION

Thus, Flow/MeMe, app was made and executed successfully. An app that spreads positiveness and encourages health was thus portrayed in an alpha version.

REFERENCES

www.udemy.com/AngelaYu

www.stackoverflow.com

www.gfg.com

www.AppIconGenerator.com

www.colourhunt.com

www.github.com

www.Assests.com

Appendix

Code

Models :

User :

```
import

struct User: Codable {

    var id  String
    var nickname: String
    var givenName: String
    var familyName: String?
    var phone  String
    var description: String?

    init id  String
         nickname  String
         givenName  String
         familyName  String
         phone  String
         description  String

        self id
        self nickname
        self givenName
        self familyName
        self phone
        self description
```

Track:

```
import
import

struct Track {

    var id   String
    var author   String
    var name   String
    var time   String
    var type   TrackType
    var image   UIImage
    var path   String
    var state : PlayingState?



enum TrackType {

    case black
    case brown
    case purple
    case grey
    case blue
```

```
    case red
    case green
    case yellow
```

AudioSession :

```
import [illegible]
import [illegible]

struct AudioSession {
    var physicalParams = PhysicalParameters()
    var emotionalParams = EmotionalParameters(moodType: .positive)
    var type     TrackType black


struct PhysicalParameters {
    var pulse  Int
    var sleep  Int
//    var pulse: Int?


struct EmotionalParameters {
    var moodType: MoodType


enum MoodType: String {
    case positive = "Positive"
    case sad     "Sad"
    case bored     "Bored"
    case inspired = "Inspiring"
    case irritable = "Irritable"
    case calm     "Calm"
    case funny     "Merry"


enum MoodSong {

    case lyric
    case happy
    case contemplative


enum SpeedType {
    case slow
```

Profile:

```
import [illegible]

struct Profile {

//    let lowerPressure: Int
    let pulse  Int
//    let upperPressure: Int
    let senderField: String
//    let mood: String
    let sleep  Int
```

View:

SampleButton :

```
import Foundation
import UIKit

class SampleButton : UIButton {

    var mainLabel    UILabel

    override init(frame : CGRect){
        super init frame frame
        self.initialSetup()
    }

    public required init   coder aDecoder NSCoder
        fatalError("init(coder:) has not been implemented")
    }

    func initialSetup(){
        self addSubview self mainLabel
        self addTarget self  action  #selector self buttonPressed   for   touchDown
        self.setupConstraints()
    }

    @objc private func buttonPressed() {
        UIView.animate(withDuration: 0.1,
            animations
                self transform    CGAffineTransform scaleX  0.9  y  0.9
            }
            completion      in
                UIView animate withDuration  0.2
                    self transform    CGAffineTransform identity
                }
            }
        }
```

ColourcollectionView:

```
import Foundation
import UIKit

class ColorCollectionViewCell : UICollectionViewCell {

    let mainView    UIView

    override func awakeFromNib() {
        super.awakeFromNib()

        self addSubview self mainView

        self.contentView.backgroundColor = .white
        self.mainView.backgroundColor = .white
        self.mainView.layer.cornerRadius = 10
```

```
        self.mainView.layer.shadowRadius = 5
        self.mainView.layer.shadowOpacity = 0.3

        self.setupSubview()
    }

    private func setupSubview() {
        self.mainView.snp.makeConstraints {
            $0.top.equalToSuperview().offset(10)
            $0.bottom.equalToSuperview().offset(-10)
            $0.left.equalToSuperview().offset(10)
            $0.right.equalToSuperview().offset(-10)
        }
    }
```

RippleView:

```
import UIKit

public extension UIView {

    func rippleBorder(location:CGPoint, color:UIColor) {
        rippleBorder(location: location, color: color, then: nil)
    }

    func rippleBorder(location:CGPoint, color:UIColor, then: ()->() ) {
        Ripple.border(view: self, locationInView: location, color: color, then: then)
    }

    func rippleFill(location:CGPoint, color:UIColor) {
        rippleFill(location: location, color: color, then: nil)
    }

    func rippleFill(location:CGPoint, color:UIColor, then: ()->() ) {
        Ripple.fill(view: self, locationInView: location, color: color, then: then)
    }

    func rippleStop() {
        Ripple.stop(view: self)
    }
```

Controller:

MainTabBar:

```
import UIKit
import SnapKit

class MainTabBarRouting {

    static func presentMainTabBarViewController(fromVC: UIViewController) {
```

```
    let vc = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "MainTabBarViewController") as!
MainTabBarViewController
    vc.modalPresentationStyle = .fullScreen
        present    animated  true  completion  nil
```

IntroductionController:

```
import Foundation
import UIKit
import SnapKit
import Firebase

class IntoductionViewController : UIViewController {

  // MARK: UIVIEW

  private let mainView    UIView
  private let startButton = SampleButton()
  private let introductionStepsView = IntroductionStepsView()

  override func viewDidLoad() {

    super viewDidLoad

    self.setupLargeTitle(title: "Music Therapy")

    self view addSubview self mainView
    self.mainView.addSubview(self.startButton)
    self.mainView.addSubview(self.introductionStepsView)
    self setupViews
    self.setupSubview()
    self.view.layoutSubviews()

    if                                nil
      LoginRouting.presentLoginViewController(fromVC: self)
    }



  // MARK: SETUP UIVIEW

  private func setupViews(){
    self.setupMainView()
    self.setupStartButton()
  }

  private func setupMainView(){
    self.view.backgroundColor = .white
    self.mainView.backgroundColor = .white
  }

  private func setupStartButton() {
    self.startButton.mainLabel.text = "Begin"
    self.startButton.mainLabel.textColor = .white
    self.startButton.backgroundColor = .black
```

```
        self.startButton.layer.cornerRadius = 10
        self.startButton.layer.masksToBounds = true
        self startButton addTarget self  action  #selector onStartClick   for   touchDown
    }

    @objc func onStartClick(sender: UIButton) {
        PhysicalParametersRouting.presentPhysicalParametersViewController(fromVC: self)
        // Add a second document with a generated ID.
    }

    // MARK: SETUP CONSTRAINTS

    private func setupSubview() {

        self mainView translatesAutoresizingMaskIntoConstraints
            edges equalToSuperview()
        }

        self startButton translatesAutoresizingMaskIntoConstraints
            left equalTo self mainView left offset 30
            right equalTo self mainView right offset -30
            bottom equalTo self mainView bottom offset self bottomHeight
            height equalTo 56
        }

        self introductionStepsView translatesAutoresizingMaskIntoConstraints
            top equalTo self mainView top offset self topBarHeight
            left equalTo self mainView left offset 30
            right equalTo self mainView right offset -30
            bottom equalTo self startButton top offset -30
        }

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .darkContent
    }

}

extension UIViewController {

    func setupLargeTitle(title : String?) {
        self.navigationController?.navigationBar.prefersLargeTitles = true
        self.navigationItem.title = title
        self.navigationController?.navigationBar.largeTitleTextAttributes = [NSAttributedString.Key.foregroundColor:
UIColor.black]
        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "", style: .plain, target: nil, action: nil)
        self.navigationController?.navigationBar.tintColor = UIColor.black
    }

}

extension UIViewController {

    var topBarHeight: CGFloat {

        if UIDevice modelName      "iPhone SE"
```

```
            return (view.window?.windowScene?.statusBarManager?.statusBarFrame.height ?? 0.0) +
                (self.navigationController?.navigationBar.frame.height ?? 30) + 30
        else
            return (view.window?.windowScene?.statusBarManager?.statusBarFrame.height ?? 0.0) +
                (self.navigationController?.navigationBar.frame.height ?? 30) + 60
    }


}

var bottomHeight: CGFloat {
    return (self.tabBarController?.tabBar.frame.size.height ?? 30) + 30
}
```

InteractiveModuleEditor :

```
import [illegible]
import [illegible]
import [illegible]
import [illegible]

class InteractiveModeViewController: UIViewController {

    // MARK: UIVIEW

    private let mainView     UIView
    private let keyboardView [illegible]
    private let containerForNote = UIView()
    private let noteLabel    UILabel
    private let swiftyOscillatorBank [illegible]
    private let containerForOscillator = UIView()

    private var delay [illegible]    nil
    private var reverb [illegible]       nil

    var track    Track

    private let screenSize  CGRect    UIScreen main bounds

    override func viewDidLoad() {

        super viewDidLoad

        [illegible]
        [illegible]
        [illegible]                self

        delay = AKDelay(swiftyOscillatorBank)
        delay   time    0.5 // seconds
        delay?.feedback = 0.2 // Normalized Value 0 - 1
        delay?.dryWetMix = 0.2 // Normalized Value 0 - 1

        reverb [illegible]          delay
        reverb [illegible]          0.5

        [illegible]                 reverb
```

```
    self.setupAudioSession()
    self.setupMainView()
    self.setupKeyboardView()
    self.setupContainerForNote()
    self.setupNoteLabel()
    self.setupOscillatorBank()
    self.setupContainerForOscillator()
    self.setupOscillator()

    self view addSubview self mainView
    self.mainView.addSubview(self.keyboardView)
    self.mainView.addSubview(self.containerForNote)
    self.mainView.addSubview(self.containerForOscillator)
    self.containerForNote.addSubview(self.noteLabel)

    // MARK: SETUP AUDIO PLOT

    let                                reverb
        plot.plotType      buffer
        plot.shouldFill    true
        plot.shouldMirror  true
        color              black
    self mainView addSubview

    self.setupSubview()


        top                self mainView      top          topBarHeight
        left               self mainView      left         30
        right              self mainView      right        30
        bottom             self mainView      centerY      60


    self.view.layoutSubviews()



// MARK: SETUP UIVIEW

private func setupAudioSession() {
    let
    debugPrint "Output volume:          "


private func setupMainView() {
    self.mainView.backgroundColor = .white


private func setupKeyboardView() {
    self keyboardView delegate    self
    self keyboardView                  true
    self keyboardView          1
    self keyboardView          3


private func setupContainerForNote() {
    self.containerForNote.isUserInteractionEnabled = true
    let                UIContextMenuInteraction delegate  self
    self containerForNote addInteraction
```

```
        self.containerForNote.layer.cornerRadius = 7
        self.containerForNote.layer.borderWidth = 2
        self.containerForNote.layer.borderColor = UIColor.white.cgColor
        self.containerForNote.layer.masksToBounds = true
        self.containerForNote.backgroundColor = .black
    }

    private func setupNoteLabel() {
        self noteLabel textColor     white
        self.noteLabel.textAlignment = .center
        self noteLabel font    UIFont systemFont ofSize  15.0
        self noteLabel text    "..."
    }

    private func setupOscillatorBank() {
//        AKManager.output = swiftyOscillatorBank
        try                  start
    }

    private func updateNoteLabel(text: String) {
        self noteLabel text
    }

    private func setupContainerForOscillator() {
        self.containerForOscillator.backgroundColor = .black
    }

    private func setupOscillator() {
//        let oscillator = AKOutputWaveformPlot.createView(width: 200, height: 250)
//        self.containerForOscillator.addSubview(oscillator)
//        oscillator.backgroundColor = .yellow
    }

    // MARK: SETUP CONSTRAINTS

    private func setupSubview() {

        self mainView
            left
            right
            top
            bottom
        }

        self keyboardView
            left          self mainView       left          20
            right           self mainView       right         20
            bottom          self mainView       bottom        20
            top           self screenSize height 2
        }

        self containerForNote
            left          self mainView       left          20
            bottom          self keyboardView       top           20
            width         30
            height        30
        }

        self noteLabel
```

```
        edges.equalTo(self.containerForNote.snp.edges)
    }

//      self.containerForOscillator.snp.makeConstraints {
//        $0.width.equalTo(200)
//        $0.height.equalTo(50)
//        $0.top.equalTo(self.mainView.snp.top)
//        $0.centerX.equalTo(self.mainView.snp.centerX)
//      }


  func makeContextMenu() -> UIMenu {
    let one = UIAction(title: "🎹", image: nil) { _ in
      self.keyboardView.octaveCount = 1
      self.keyboardView.reloadInputViews()
      self.keyboardView.setNeedsDisplay()
    }
    let two = UIAction(title: "🎹🎹", image: nil) { _ in
      self.keyboardView.octaveCount = 2
      self.keyboardView.reloadInputViews()
      self.keyboardView.setNeedsDisplay()
    }
    let three = UIAction(title: "🎹🎹🎹", image: nil) { _ in
      self.keyboardView.octaveCount = 3
      self.keyboardView.reloadInputViews()
      self.keyboardView.setNeedsDisplay()
    }
    let four = UIAction(title: "🎹🎹🎹🎹", image: nil) { _ in
      self.keyboardView.octaveCount = 4
      self.keyboardView.reloadInputViews()
      self.keyboardView.setNeedsDisplay()
    }

    return UIMenu(title: "Октавы", children: [one, two, three, four])
  }

  override func becomeFirstResponder() -> Bool {
    return true
  }

  override func motionEnded(_ motion: UIEvent.EventSubtype, with event: UIEvent?) {
    if motion == .motionShake {
      print("Shake Gesture Detected")
    }
  }

  override func viewDidDisappear(_ animated: Bool) {
    super.viewDidDisappear(animated)
    do {
      try conductor.engine.stop()
    } catch {
      fatalError("AudioKit crash")
    }
  }
```

```
extension InteractiveModeViewController: AKKeyboardDelegate {

    func noteOn note MIDINoteNumber) {

        debugPrint recursiveNote Int    magnitude    rawValue

        let                    recursiveNote Int      magnitude    rawValue
        self updateNoteLabel text
        self swiftyOscillatorBank                                  80
    }


    func noteOff note MIDINoteNumber) {
        self swiftyOscillatorBank
    }



extension InteractiveModeViewController : UIContextMenuInteractionDelegate {
    func contextMenuInteraction(_ interaction: UIContextMenuInteraction, configurationForMenuAtLocation location:
CGPoint) -> UIContextMenuConfiguration? {
        return UIContextMenuConfiguration identifier  nil  previewProvider  nil  actionProvider                in
        return self.makeContextMenu()
        }
    }



extension InteractiveModeViewController : AKMIDIListener {

    func receivedMIDINoteOn(noteNumber: MIDINoteNumber,
                            velocity
                            channel
                            portID                nil
                            offset                0
        print "note                    on"
            DispatchQueue.main async
                self noteOn note
            }
        }


    func receivedMIDINoteOff noteNumber                velocity              channel                portID
                offset
        print "note                    off"
            DispatchQueue.main async
                self noteOff note
```