Software Design Specification Document
(CS360)

Guftaar

Group Number: 04

Saad Sher Alam
Emaan Atique
Harris Ahmed
Bakhtawar Ahtisham
Emaan Bilal
Romessa Shah Jahan

Course: Software Engineering CS360

Instructor: Maryam Abdul Ghafoor

University: Lahore University of Management Sciences (LUMS)

Version: 1.0

Date: (16/03/2020)

Number of hours spent on this document: 200

CONTENTS

1 Change Log

1.1 Project Scope

Guftaar is intended to be an English language m-Health web application catering to PWS (people who stutter), with virtual multi-feature treatment support, detailed in the following section. The application is meant to be an online platform providing at-home self management and professional assistance to those diagnosed, with instructional training sourced from established speech therapists, and peer-to-peer support via online interactions. Guftaar end users include both PWS and coaches along with the system administrators, whose roles and app involvement is explained in the table below (users listed in order of importance- based on frequency of use and level of interaction with the portal).

End-User	Function
PWS	Access guided training, course material and auditory support with progress measures, community support via live sessions and certified medical help with in-app incentive mechanisms
Coaches	Connect with PWS according to availability to deliver virtual support via online sessions and facilitate speech therapy via feedback logging and progress monitoring
Administrator	Manage existing resources, profiles and content hosted on the application along with maintaining the application database and monitoring in-app activity

Table 1: End-User Descriptions

1.2 Change log

The following changes have been made to Guftaar since the SRS:

- Instead of only being able to add coaches, admin will also be able to add other admins to the Guftaar platform. Hence, the use case has been renamed to '**Add Employees**.' The sign-up option is only available to the clients, as specified by the SRS document.
- The blog functionality has been removed from the Guftaar platform. Instead, Guftaar intends to focus on independent and guided speech support through daily activities and quick practice, sessions with coaches, and courses.
- The number of options for **'Daily Assessment'** have been reduced to 3: No Stuttering, Moderate and Extreme.
- The number of options in the **'Syllable Counting'** activity under daily practice has been reduced from 4 to 3 for ease of understanding and usability.
- Admins can now view the total number of clients, coaches and admins using the platform.
- Admins can now view all coaches and also have a categorization of top rated coaches.
- Coaches will also get meeting reminders for all upcoming sessions.

2 Introduction

2.1 Document Purpose

This document is meant to detail the complete software design specifications as part of the Software Requirement Specifications for Project Guftaar, the semester-long project for the CS 360: Software Engineering course at the Lahore University of Management Sciences. Guftaar is a web application that provides virtual multi-feature treatment support, designed for PWS (people who stutter) in the English language.

This document offers a comprehensive overview of our internal system, including an in-depth description of the various components and subsystems that govern the platform, as well as how they are interconnected. We use activity and sequence diagrams to illustrate the internal flows, which delineate how processes are executed from the moment users choose an option to when they obtain the desired outcome. Additionally, we delve into the high-level architectural system and database design we have adopted to provide a more thorough understanding of Guftaar's backend system design. Alongside the internal requirements and design specifications, this document also covers the external user interface, with a particular focus on its design and layout.

2.2 Product Scope

In light of the impact a communication deficit can create in the lives of those affected, this project is meant to serve as a technological intervention for young adults that supports speech and language therapy in the form of a web application. The platform is intended to empower those struggling with articulation with an opportunity to manage their fluency through an interactive, easy to use interface that provides resources in the form of exercises, progress tracking, live sessions as well as a way to connect with certified instructors for individual training, and course sign ups.

In the study of existing literature, and through PWS interaction, the need for a holistic technological support system was validated as a necessary shift from traditional speech therapy in the country. Conventional support was quoted to be expensive, ranging from PKR 2000 to PKR 5000 per session, outdated, prone to relapse (80% occurrence) and laborious for all parties involved [1]. Resources also appeared scattered, progress unregulated and access to certified help difficult to reach. Subsequently, a tech-enabled platform where end users could find a host of resources to assist their personal speech improvement journey, and access existing support mechanisms in the form of experienced professionals is an essential layer of help currently absent in the local market. Guftaar is a project meant to accomplish just that.

2.3 Intended Audience and Document Overview

This document provides a detailed overview of Project Guftaar, including its system division, architectural choices, user interface design and database choice. As a result, it serves as a comprehensive guide for developers to follow, plan and monitor their progress using the specification details and internal flows of use cases as well as the expected output (as shown by the external user interface designs). The project lead and advisor can also use this document to fully understand the project's scope and software specifications to manage resources efficiently, such as workers, time, and finances. Additionally, our client (the course staff) can refer to this document to assess the completeness of the project at the end of the semester and compare the final product with the user interface promised/specified in section 5 of this document.

To start with the development process of the Guftaar system, developers should begin by referring to Section 4, which provides a detailed overview of the subsystems to better understand how each functionality will be implemented and interlinked. Next, they should carefully review the detailed high level architectural specifications, also given in Section 4, to plan their implementation accordingly. In addition, Section 4 should also be consulted to understand the database choice, design and detailed table components before setting up the backend system. For testers, course staff, and clients, it is recommended that they read this document carefully to fully understand the software design and architecture of the Guftaar system. Then they can compare these promises with the final product deployed by the team to make a fair evaluation. To understand any technical terms used in this document, Section 2.4 for definitions can be referred to. For managers, leaders, and advisors, it is suggested that they read this document thoroughly in the given order to have a complete understanding of the project backend and frontend design choices.

2.4 Definitions, Acronyms and Abbreviations

Term	Definition
PWS	Access guided training, course material and auditory support with progress measures, community support via live sessions and certified medical help with in-app incentive mechanisms
Coaches	Connect with PWS according to availability to deliver virtual support via online sessions and facilitate speech therapy via feedback logging and progress monitoring

Administrator	Manage existing resources, profiles and content hosted on the application along with maintaining the application database and monitoring in-app activity
m-Health	A term used for the practice of medicine and public health which is supported by mobile devices.
Javascript	A high-level programming language that is widely used for both front-end and back-end web development. It is a client-side scripting language that runs in a user's web browser and is used to create interactive web pages, web applications, and mobile apps.
ReactJS	It is and open source Javascript library for building user interfaces
Backend	Backend refers to the server-side of an application or website, which includes the server, database, and application logic that enable the frontend (client-side) to interact with the user and respond to their requests.
Frontend	Frontend refers to the part of a website or application that the user interacts with directly. It typically includes the visual design, layout, and user interface elements, as well as any functionality that is provided to the user
Sequence Diagram	A sequence diagram is a visual representation of the interactions between objects in a system. It shows the order in which messages are exchanged between the objects and can be used to analyze and design complex systems.
Component Diagram	A component diagram is a UML (Unified Modeling Language) diagram that shows the organization and dependencies of the components in a system. It depicts how the components interact with each other to achieve a specific functionality or service.

Figma	Figma is a cloud-based design and collaboration tool used by designers, developers, and product managers to create and share user interfaces, prototypes, and design systems. It allows real-time collaboration, version control, and easy sharing of design files.
Visual Studio Code	Visual Studio Code is a free and open-source code editor which provides a wide range of features, including code highlighting, debugging, source control integration, and extensions for various programming languages.
MVC Presentation Layer	The MVC (Model-View-Controller) presentation layer is a design pattern used in software development to separate the application's user interface (View) from the business logic (Controller) and data storage (Model). The Model represents the data and business logic, the View displays the data to the user, and the Controller handles user input and updates the Model and View accordingly.
Peer-to-Peer Support	Refers to a type of support where individuals with similar experiences provide assistance, advice, and emotional support to each other. This support can occur in person or through online platforms.
Speech Therapist	A healthcare professional who specializes in diagnosis, treatment, and prevention of speech and language disorders.
Speech Pathologist	Another term for a speech therapist
Speech Disorders	Conditions that affect an individual’s ability to produce speech sounds correctly or fluently, or to understand or use language.
User-centered design	An approach to designing products or services

	that focuses on the needs, wants and limitations of the end user.
Server	A computer or program that provides data or services to other computers or programs.
Hosting	Process of storing and managing a website or application on a server that is accessible through the internet.
Latency	Amount of time it takes for a request to be sent from client to a server and for the response to be received back.
Interface/ protocol requirements	Technical specifications that must be met in order for two systems or devices to communicate with each other.
Asynchronous Programming	A programming model that allows multiple tasks to be executed concurrently without blocking other tasks.
RESTful APIs	An application programming interface that uses HTTP requests to GET, POST, PUT, and DELETE data.
Firebase or Pocketbase	Cloud-based platforms that provide backend services for web and mobile applications
NoSQL	A type of database that does not use the traditional relational model used by SQL databases
WebSocket communication	A protocol that enables real-time communication between a client and a server over a single TCP connection.
ClientSphere	A subsystem in the website that is responsible for maintaining client facing interactions with the Guftaar application
Sub-system	A smaller system that is part of a larger system

Sub-subsystem	An even smaller system that is a part of a slightly larger system which is in turn part of a larger system
Client	A computer program or device that requests services or data from a server
NodeJS	An open-source, cross-platform JavaScript runtime environment that executes JavaScript code on the server-side.
DOM	stands for Document Object Model, which is a programming interface for web documents that allows them to be manipulated and modified.
API	stands for Application Programming Interface. It is a set of protocols, routines, and tools for building software applications.
MongoDB Atlas	cloud-based fully managed database service that provides automatic scaling, backup and recovery.
MySQL	A popular open-source relational database management system.
Zoom	Video conferencing software that allows users to conduct online meetings, webinars, and virtual events.
GUI	stands for Graphical User Interface. It is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.
Tailwind	A utility-first CSS framework that allows for rapid UI development.
CSS	Stands for Cascading Style sheet. It is a style sheet language used for describing the presentation of a document written in HTML or XML.

Brotli compression and PurgeCSS	These are the tools used to optimize website performance by compressing and removing unused CSS code respectively.
Information Architecture	It provides an operational map to how a product acts and functions work for users.
React MUI library	Collection of pre-built React components that can be used to rapidly build user interfaces.
UI	User Interface is the space where interactions between humans and machines occur.
Usability	It refers to the degree to which a product or system can be used by its intended users to achieve their goals effectively, efficiently, and with satisfaction.
Streak	Streak count generally refers to the number of consecutive occurrences of a particular event or behavior. It is commonly used in the context of tracking and measuring progress or consistency over time.
Linklater Voice Progression activity	A speech therapy technique that involves a series of exercises designed to improve vocal control and flexibility
Syllable	It is a unit of sound that forms a word or part of a word. It is made up of one or more vowel sounds and may also include consonant sounds that come before or after the vowel sound.
NavBar	A NavBar (short for Navigation Bar) contains a set of clickable links or buttons that allow users to navigate to different sections or pages within the website or application.
Dashboard	A graphical user interface that provides a summary of key performance indicators or other information.

Calendly	It is a popular cloud-based scheduling software tool that allows users to schedule and manage appointments, meetings, and events.
HTML	stands for Hypertext Markup Language. It is a standard language used for creating web pages and other online content.
SQL Injection Attacks	A type of cyber attack where an attacker uses malicious SQL code to gain access to a database or manipulate its contents.
Third Party Source	Any software, application or service that is not owned or controlled by the organization or individual using it.
Malicious Attack	An intentional attempt to damage, disrupt, or gain access to a database or manipulate its contents.
Nielson's and Norman Design Principles	A set of design principles created by Jakob Nielson and Don Norman that aim to improve the usability and user experience of digital products and services.
WordPress	A popular content management system used for creating and managing websites
Hashed/Encrypted Password	A password that has been transformed into a complex string of characters using a mathematical algorithm to protect the user's identity and secure their account.
Google Drive	A cloud-based file storage and synchronization service provided by Google.l
APM	It stands for Application Performance Management and refers to the practice of monitoring and managing the performance of software applications in production environments.

Parameterized SQL	It refers to a method of executing SQL queries within a computer program by passing parameters or variables to the query rather than embedding them directly in the query string.
Windows, Mac OS X, iOS, and Android	Operating Systems that manage a computer's hardware and provide a platform for running other software applications.
Bootstrap	It refers to the process of loading and initializing a computer system or software program.
Deployment	It refers to the process of making a software application or system available for use in a production environment
Source Code	It refers to the set of instructions written in a programming language that can be read and understood by humans, and which are then compiled or interpreted into machine-executable code that a computer can understand and execute.

Table 2: Definitions

2.5 References and Acknowledgments

[1] "In Pakistan, Lack of Trained Speech Therapists Feeding Quackery | the Express Tribune." *The Express Tribune*, 21 Oct. 2019, tribune.com.pk/story/2083912/pakistan-lack-trained-speech-therapists-feeding-quackery.
[2] Vilmate. (2023, February 22). *Why choose react for front-end development in 2021*. VILMATE. Retrieved March 14, 2023, from https://vilmate.com/blog/why-choose-react-in-2021/#:~:text=Accordingly%2C%20React%27s%20advantage%20over%20other,changed%20in%20the%20Real%20DOM
[3] Technologies, E. (2022, December 16). 6 reasons why tailwind CSS is worth it. Retrieved March 14, 2023, from https://encircletechnologies.com/blog/advantages-of-tailwind-css/#:~:text=One%20of%20the%20main%20advantages,%2C%20styling%2C%20themes%2C%20etc
[4] Aela, E. (2022, June 29). Nielsen's heuristics: 10 usability principles to improve UI design. Retrieved March 15, 2023, from https://aelaschool.com/en/interactiondesign/10-usability-heuristics-ui-design/
[5] What are Norman's design principles? (n.d.). Retrieved March 15, 2023, from https://www.educative.io/answers/what-are-normans-design-principles

3 Overall Description

3.1 System overview

Functionality

Guftaar is a computer-based application that helps speech pathologists and therapists in assessing and treating speech disorders. The software system is designed to provide a comprehensive solution that can cater to different speech disorders, including articulation, fluency, voice and language disorders. The speech therapy software system offers a wide range of functionalities that are essential for people who stutter (PWS) in their daily practice. Some of the key features of the software system include:
Assessment tools: The software system provides a variety of assessment tools, including standardized tests and customizable assessments, to evaluate the speech and language abilities of patients. **Treatment planning**: Based on the results of the assessments, the software system generates a customized treatment plan that includes various exercises and activities to improve the patient's speech and language abilities. **Progress tracking**: The software system enables therapists to monitor the progress of their patients by tracking their performance on various exercises and activities. **Reporting:** The software system generates detailed reports that provide a summary of the patient's progress and performance, which can be shared with the patient's caregivers and other health care providers. They also have an option to share their reports with therapists online who are working with Guftaar.

Design Approach

The design of the speech therapy software system is based on a user-centered approach. The system is designed to be intuitive and easy to use, even for therapists who may not be technologically savvy. The system's interface is designed to be visually appealing and user-friendly, with clear and concise instructions.

Guftaar is also designed to be flexible, allowing therapists to customize the system to meet their specific needs. The system provides a range of customization options, including the ability to create custom assessments and treatment plans.

Organization

The system is organized in such a way as to allow the user to progress through a series of levels, starting with basic exercises and gradually moving on to more advanced ones. The exercises are presented in a variety of formats, including text, audio, and video, to cater to different learning

styles. The system also includes feedback mechanisms that allow the user to track their progress and receive guidance on areas that require improvement.

Overall, the speech therapy software is a comprehensive solution that provides therapists with the tools they need to assess and treat speech disorders. Its user-centered design and flexible customization options make it an essential tool for both a person who stutters and a speech pathologist.

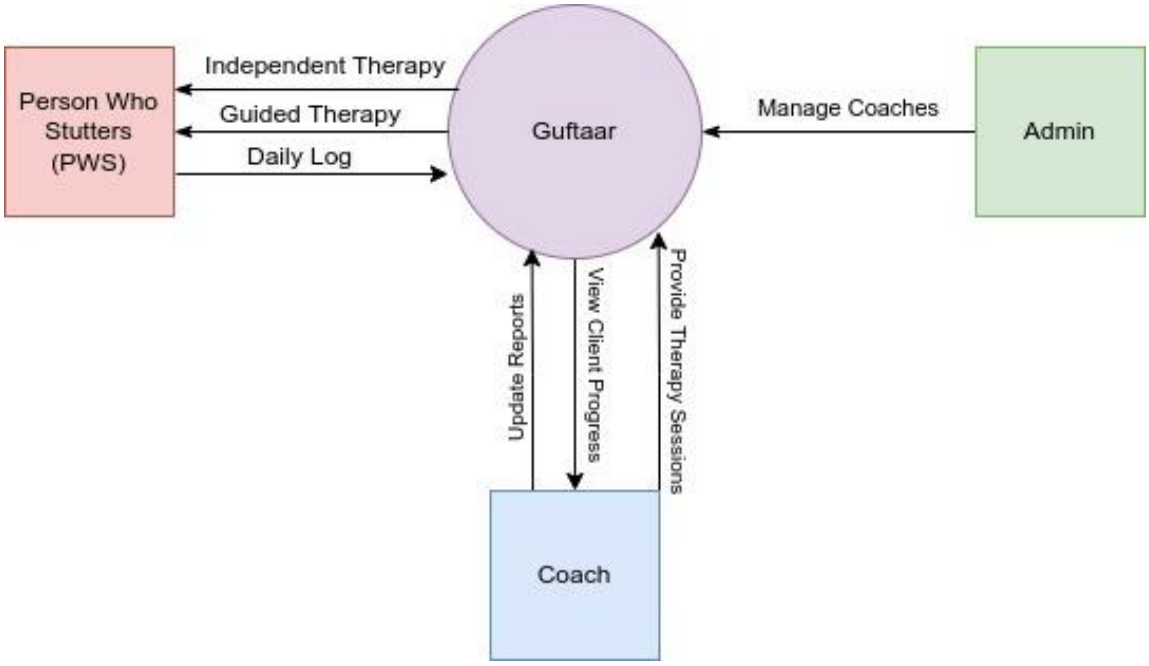


Figure 1: Context Diagram

3.2 System constraints

Schedule: The project is constrained by a deadline with a development period of roughly three months. As a result, Guftaar might not be able to fully incorporate automated interactive activities which include hosting the application on multiple servers. Functionalities that require single server deployment will be incorporated.

Budget: For production and development purposes, Guftaar utilizes cost-free platforms such as Firebase or Pocketbase. For deployment, subscriptions for softwares to be reused, hosting platforms, and other services will be taken into account.

Resources: Some services for Guftaar require speech-language pathologists and certified speech therapists/coaches. To make a fully-functional version of Guftaar, we will be utilizing open-source data for these services, such as courses and activities provided by already established speech therapy institutions. On the other hand, other features are completely automated with response and feedback integration and won't require third-party data.

Hardware and Software Environment: The software's design must be compatible with the hardware and software environment on which it is intended to run. This includes considerations such as operating system, processing power, memory capacity, and storage capacity. The software must be optimized to work effectively with the available resources to ensure optimal performance and user experience.

End-User Environment: The software must be designed with the end-user environment in mind. This includes factors such as the user's age, education level, language proficiency, cognitive abilities. The software must be easy to use, intuitive, and accessible to all users, regardless of their level of technical expertise.

Standards Compliance: The software must be compliant with industry standards and regulations. Compliance with standards ensures interoperability with other systems, compatibility with hardware and software, and adherence to best practices in the field. Examples of such standards include HIPAA, and GDPR.

Security Requirements: The software must adhere to strict security requirements to protect sensitive data and ensure the privacy and confidentiality of users. This requires the use of encryption, secure authentication, and access controls to prevent unauthorized access or breaches.

Performance Requirements: The software must meet performance requirements to ensure that it runs efficiently and provides a seamless user experience. This includes optimizing the software to minimize resource utilization and reducing latency to ensure that it responds quickly and consistently.

Verification and Validation Requirements: The software must adhere to strict verification and validation requirements to ensure that it meets its intended purpose and functions as expected. This requires extensive testing and quality assurance processes to identify and resolve any issues or bugs that may impact the software's functionality or performance.

Other Constraints: Other constraints that may impact the design of software for speech therapy include network communications such as a working Wireless connection or 3G/4G mobile connection, interoperability requirements, interface/ protocol requirements, and other requirements described in the requirements specification.

To conclude, the design of software for Guftaar is subject to various global limitations and constraints that affect its development, performance, and functionality. By carefully considering these constraints and designing the software to meet them, developers can ensure that their software provides an optimal user experience and meets the needs of its intended audience.

3.3 Architectural strategies

The design of the speech therapy system is influenced by several factors, including the choice of programming language, reuse of existing software components, and plans for future enhancements.

Programming Language

The system is designed using JavaScript due to its simplicity, readability, and availability of enriched package managers (npm and yarn) that support natural language processing and audio processing. JS also allows for integration with other languages, making it easier to extend the system in the future.

Reuse of Existing Software Components

To accelerate the development process and reduce errors, we leveraged existing libraries and frameworks such as NLP.js for natural language processing, Firebase by Google for backend development, and NoSQL for database management. These components are open-course and well-tested, reducing the risk of errors and improving the overall quality of the system.

Future Plans for Extending or Enhancing the Software

The system is designed with extensibility in mind, allowing for future enhancements as required. For example, we plan to integrate machine learning algorithms to improve the accuracy of speech recognition and provide personalized feedback to users. We also plan to add support for different languages to cater to users from diverse backgrounds.

User Interface Paradigms

Since we’re using firebase as part of our backend, Guftaar uses NoSQL as the primary database management system, providing a robust and scalable solution for storing and retrieving user data. The system also employs memory management policies to optimize system performance and reduce memory usage.

Data Management Policies

The system uses PostgreSQL as the primary database management system, providing a robust and scalable solution for storing and retrieving user data. The system also employs memory management policies to optimize system performance and reduce memory usage.

Distributed Data and Control

The system is designed to support distributed data and control over a network, allowing multiple users to access the system simultaneously. This is achieved through the use of RESTful APIs, which provide a standardized mechanism for communicating between different components of the system.

Concurrency and Synchronization

To ensure the system is highly responsive and can handle multiple requests concurrently, we use asynchronous programming techniques in combination with task queues. This allows the system to perform background tasks while responding to user requests in real-time.

Communication Mechanisms

The system uses WebSocket communication to provide real-time feedback to users during their therapy sessions. This allows the system to provide immediate feedback to users, improving the overall effectiveness of the therapy process.

In conclusion, the design of the speech therapy system is influenced by several factors, including the choice of programming language, reuse of existing software components, and plans for future enhancements. The system is designed with extensibility, scalability, and user-friendliness in mind, providing a robust and effective solution for speech therapy.

4 System Architecture

4.1 System Architecture

4.1.1 System Decomposition/Components

We employed a bottom-up methodology for decomposing our system, beginning with the functional requirements and use cases specified in our SRS document. This allowed us to design a three-stage decomposition plan.
Analyzing the flows and processes of each use case (what it required, what processes it included and what it achieved), we first grouped together our system into the following 8 categories:

1. **Speech Therapy Content Management**: This sub-subsystem is responsible for managing speech therapy content, such as daily activity tasks (question bank and answer options for syllable counting), speech techniques (words and their corresponding audio links), and quick practice paragraphs. Based on the daily task activity progress, the system also monitors the daily task streak. All in all, this sub-subsystem constitutes the following use cases:
    a. Quick Practice
    b. Daily Tasks and Activities
    c. Daily Task Streak
    d. Speech Techniques

2. **Audio Management**: This sub-subsystem is responsible for recording, analyzing and processing speech input from clients under the quick practice activities.

3. **Progress Tracking**: This sub-subsystem is responsible for tracking user progress and providing feedback to users by generating reports on user performance. It therefore includes the following use case:
    a. Generate Progress Reports
    b. Share Progress Reports with Coach

4. **Scheduling**: This sub-subsystem is responsible for facilitating communication between users and therapists such as updating coach availability, scheduling meetings and giving meeting reminders. It includes the following use cases:
    a. Scheduled a Session
    b. Send Meeting Reminders

5. **Payment and Billing**: This sub-subsystem is responsible for managing user payments and billing for the premium speech therapy courses. It includes the following use case:

   a. Buy a Course

6. **Coach Management:** This subsystem is responsible for maintaining and updating coach details to better facilitate the following functionalities:
   a. Mark Meeting Availability
   b. Add Client Notes
   c. View Client Notes

7. **Admin Logistics:** This subsystem is responsible for managing admin privileged tasks such as:
   a. Creating Coach Accounts
   b. Adding Admins
   c. Updating Coach Ratings

8. **Client Sphere:** This subsystem is responsible for maintaining client facing interactions with the Guftaar application, like:
   a. Logging Daily Progress
   b. Displaying Encouraging Strength Statements
   c. Collection of Client Feedback on Coaches.

Next, we grouped together the sub-subsystems based on the high level purpose that each fulfilled. This decomposes the Guftaar system into the following 5 subsystems each including the enlisted sub-subsystem/Use Cases:

1. **User Management:**
   a. Admin Logistics
   b. Coach Management
   c. ClientSphere

2. **Authentication:**
   a. Log In
   b. Log Out
   c. Sign Up
   d. Update Password

3. **Independent Speech Therapy:**
   a. Audio Management
   b. Speech Therapy Content
   c. Progress Tracking

4. **Database**
5. **Guided Therapy:**
    a. Scheduling
    b. Payment and Billing

This decomposition of our system can be visualized in the following diagram:

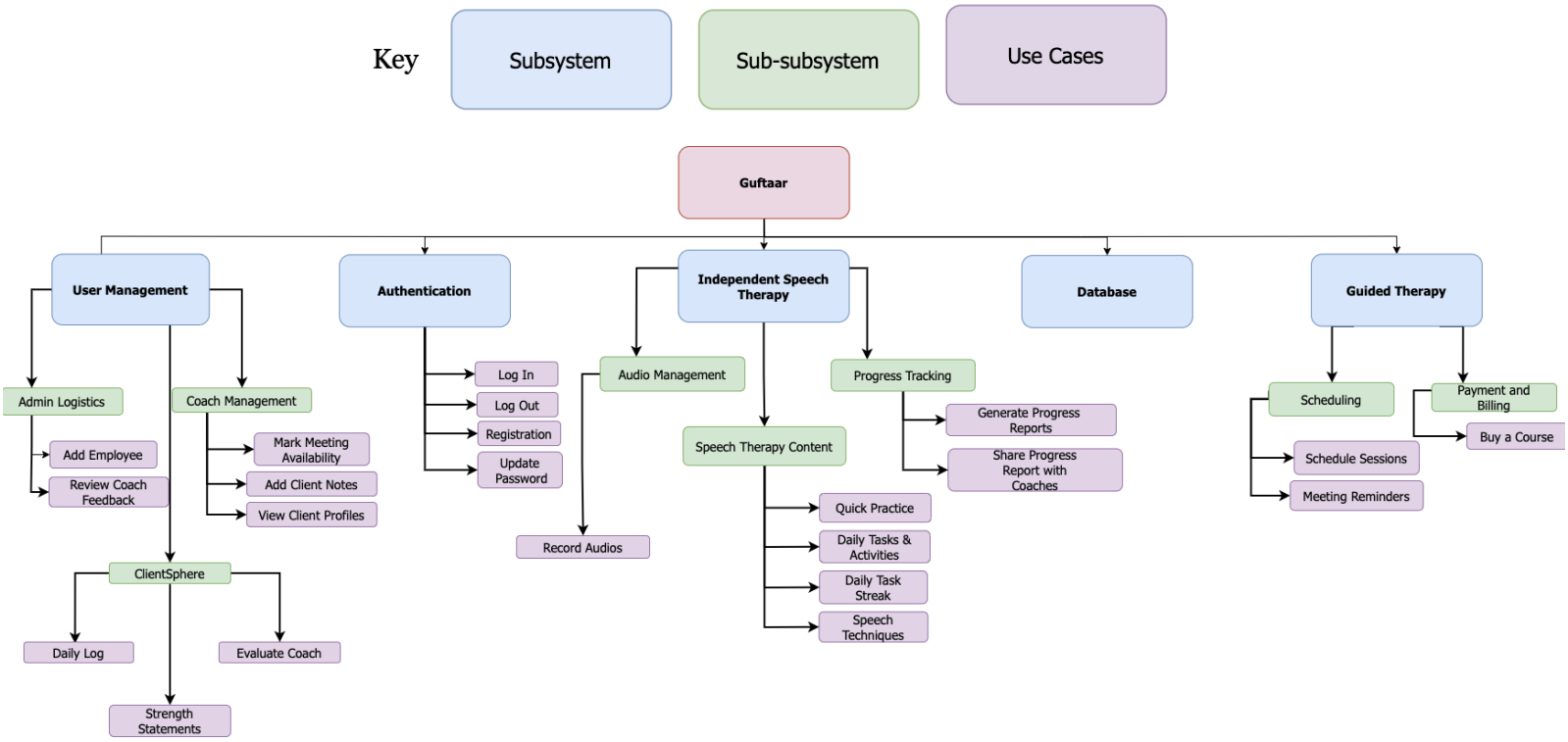


Figure 2: System Decomposition

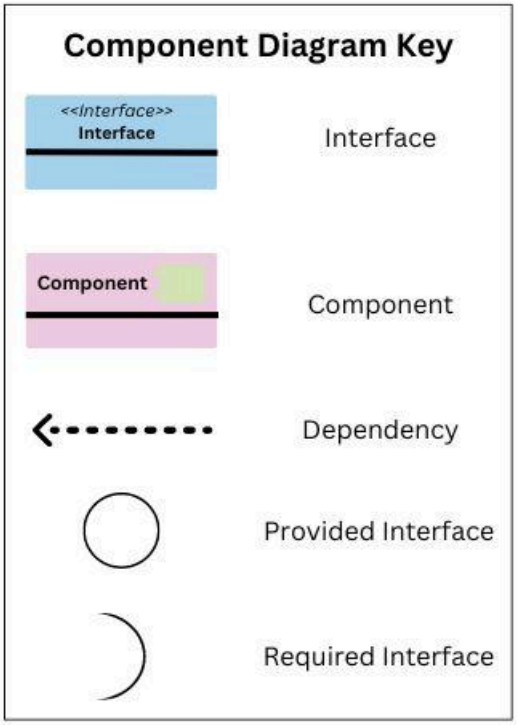
The different subsystems are interconnected and work together to provide a seamless experience for the clients and coaches alike, from registration and authentication to progress tracking and billing. For example:

1. The Authentication subsystem interacts with almost all other subsystems as it is responsible for verifying users and determining their level of access within the system.
2. The Database subsystem interacts with all other subsystems for data retrieval.
3. The Progress Tracking sub-subsystem and the Speech Therapy Content sub-subsystem interact when generating progress reports for clients. The Progress Tracking sub-subsystem

retrieves data on the client's performance from the Speech Therapy Content sub-subsystem, and uses this data to generate a report for the client to view.

4. The Scheduling sub-subsystem interacts with the Coach Management sub-subsystem to allow clients to select a coach, view their availability and schedule a meeting.
5. The Speech Therapy Content sub-subsystem interacts with the Audio Management sub-subsystem to rate the user progress for the audio uploaded in the quick practice activity. The Speech Therapy Content sub-subsystem selects the text to be read by the client, and the Audio Management sub-subsystem records the client's voice and analyzes their speech to provide feedback on their performance.
6. The Coach Management sub-subsystem may interact with the Progress Tracking sub-subsystem to provide client progress reports to the therapists if allowed by the client.
7. Admin Logistics sub-subsystem interacts with the coach management sub-subsystem to create coach profiles and update their ratings.
8. ClientSphere interacts with the Admin Logistics sub-subsystem so a system is created where client feedback is valued with the thorough feedback review process.
9. ClientSphere and Payment and Billing sub-subsystems interact so that clients can successfully access premium courses.

These interactions between different sub-subsystems can be viewed in the following component diagram:



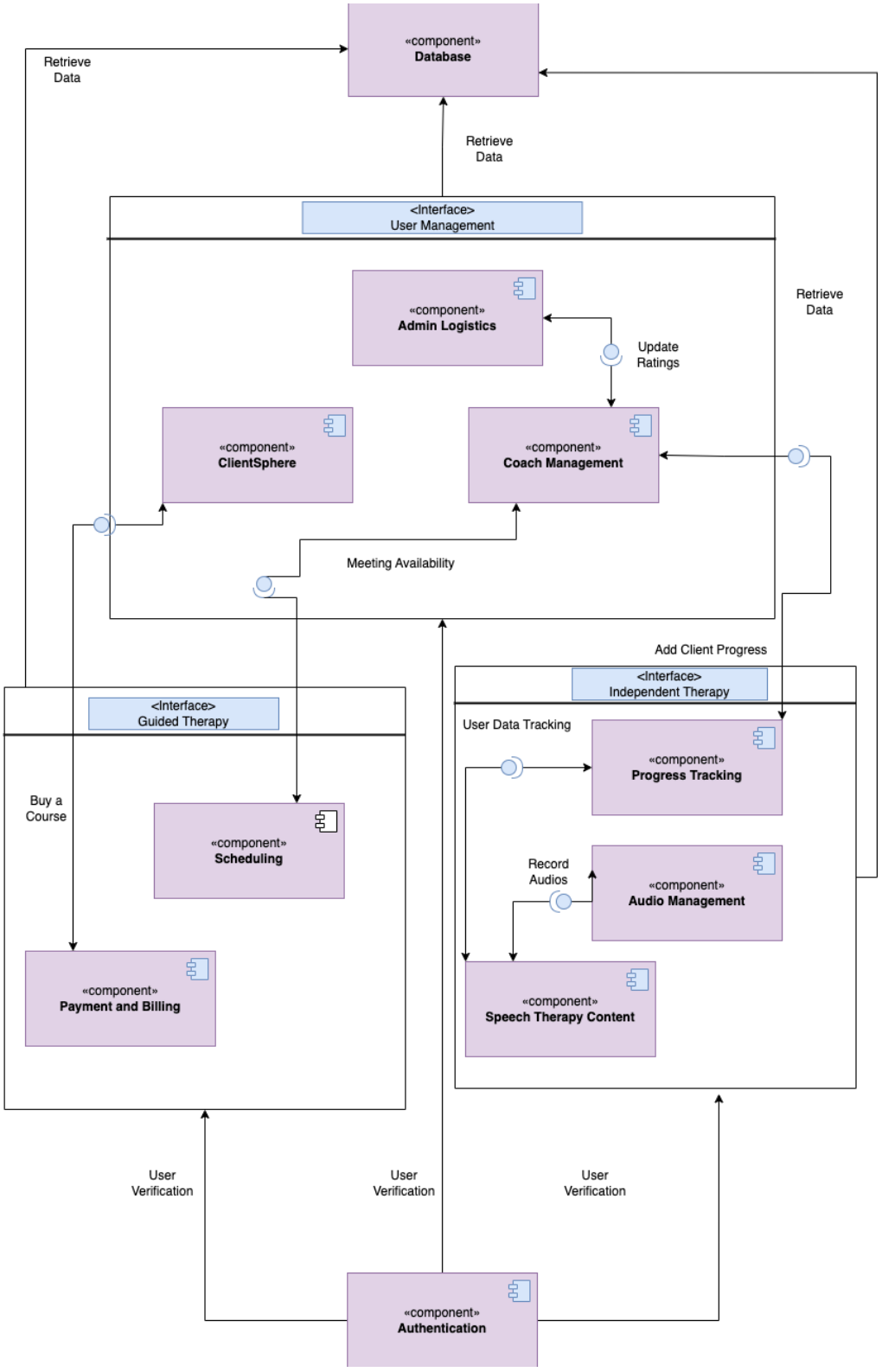
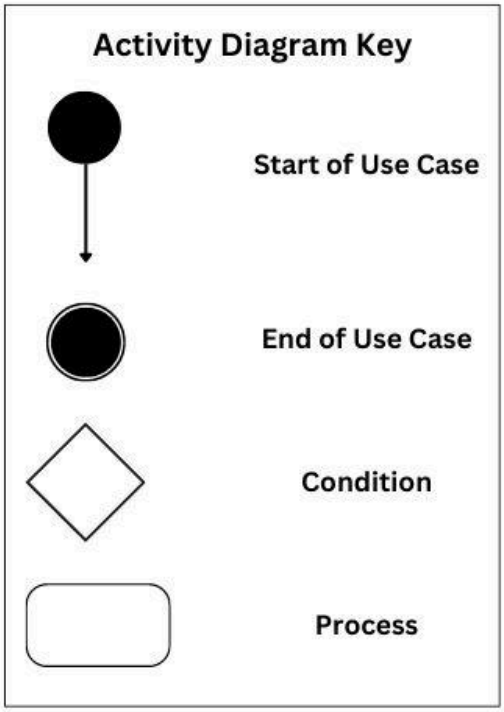


Figure 3: Component Diagram

4.1.2 Internal Flows

The following Activity Diagram shows the detailed flow of processes that are executed for our three main use cases:

1. Add Coach
2. Add Meeting Availability
3. Daily Activities



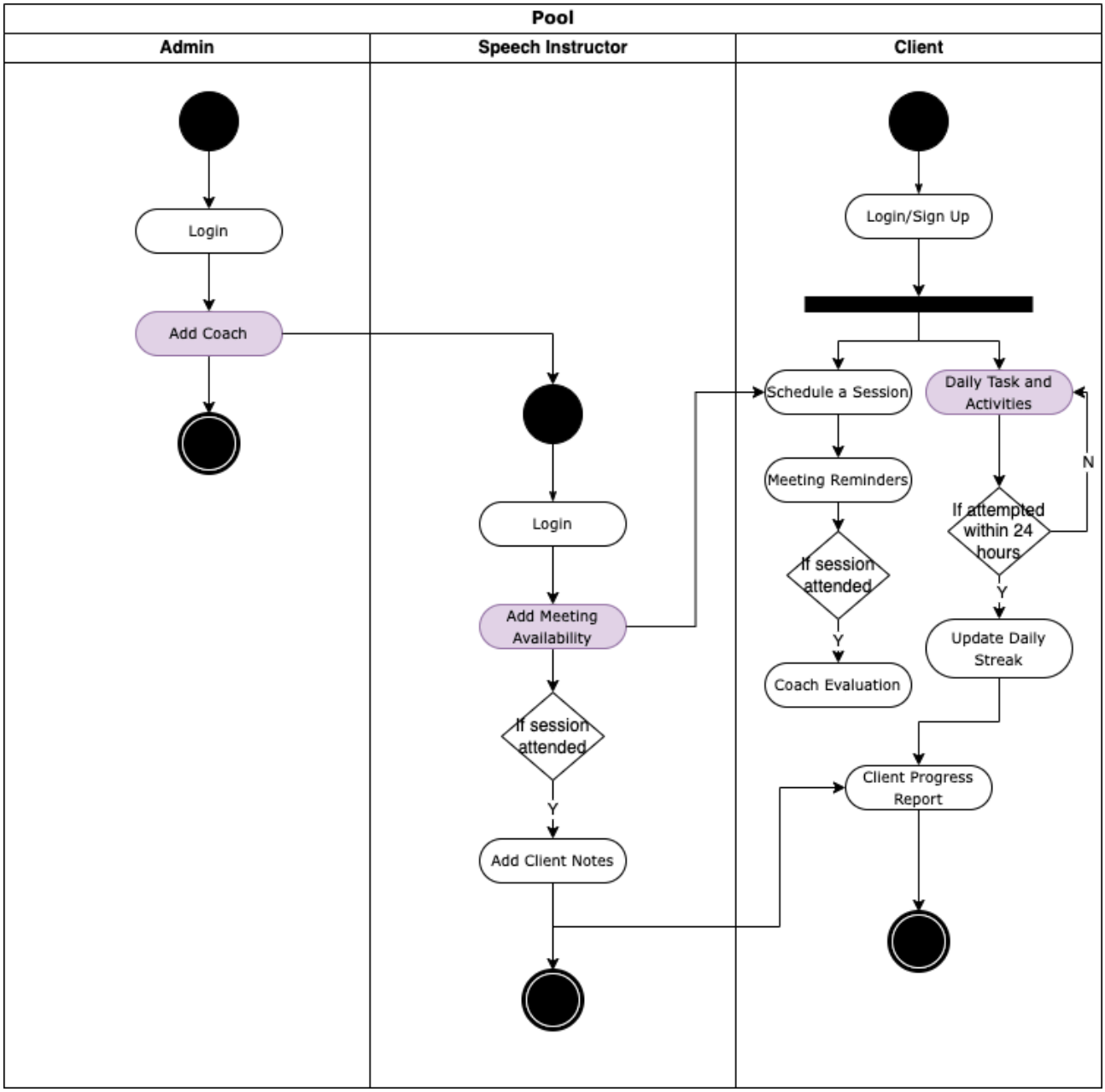


Figure 4: Activity Diagram

4.1.3 High Level Architecture

Guftaar uses a client-server model with a combination of the MVC (Model, View, Controller) architecture. In such an application, the client-side code (View and Controller) runs on the user's browser, while the server-side code (Model) runs on a remote server. This approach offers several advantages, including:

- Improved performance: By offloading some of the processing to the client-side, the server can handle more requests and reduce the load on the server.
- Improved scalability: The client-server model allows the application to scale horizontally by adding more servers to handle the increased load.
- Improved security: The separation of the client-side and server-side code helps to prevent unauthorized access to sensitive data.

Our speech therapy software system is built using ReactJS for the frontend, NodeJS for the backend, and MongoDB Atlas for the database, which is hosted on the cloud. ReactJS is used because it allows for easy management and rendering of the user interface. ReactJS uses a virtual DOM, which makes it faster and more efficient than other JavaScript libraries. We used NodeJS for the backend because it too allows for easy management of server-side resources and APIs. NodeJS also has a vast library of packages and modules that can be easily integrated into the application. The database for this application is hosted on MongoDB Atlas, which makes it easy to manage and scale the database as per the application requirements. MongoDB Atlas also provides several features like automated backups, point-in-time recovery, and security features like encryption and authorization.
In conclusion, this web application is built using ReactJS for the frontend, NodeJS for the backend, and MongoDB Atlas for the database.

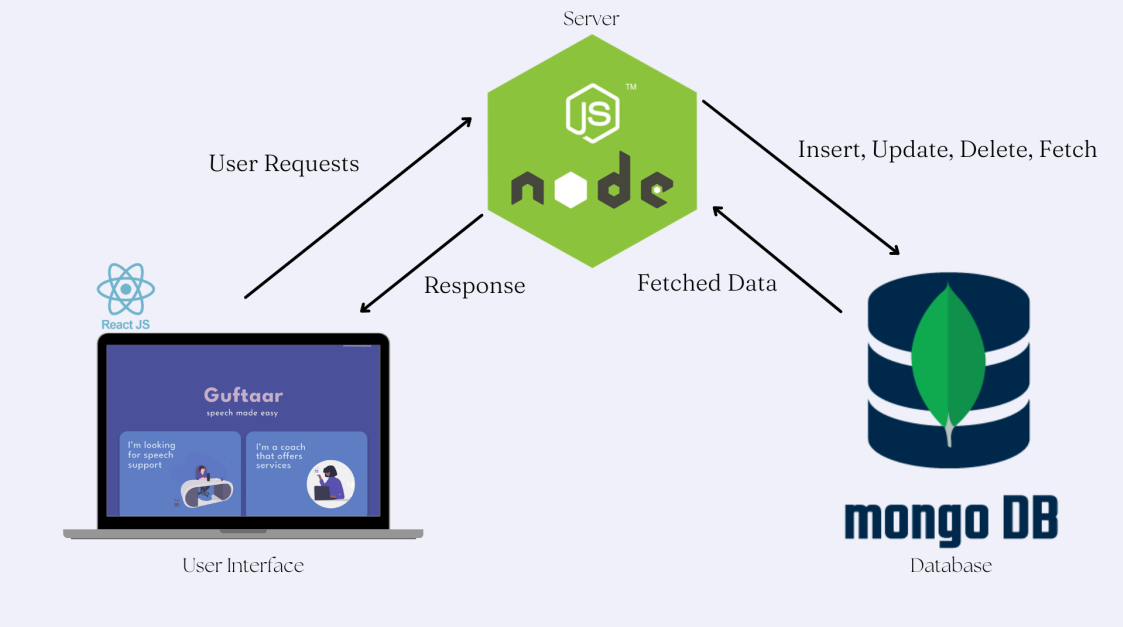


Figure 5: High Level Architecture

4.2 Subsystem Architecture

To detail out our software architecture more, in this section we first provide sequence diagrams for our top three use cases:

1. Add Coach
2. Add Meeting Availability
3. Daily Tasks and Activities
    a. Syllable Counting
    b. Breathing Exercises
    c. Linklater Voice Progression

These sequence diagrams are meant to give a deeper insight to the interactions between the different architectural components (User Interface, Server, Database) for each of the use cases mentioned above.

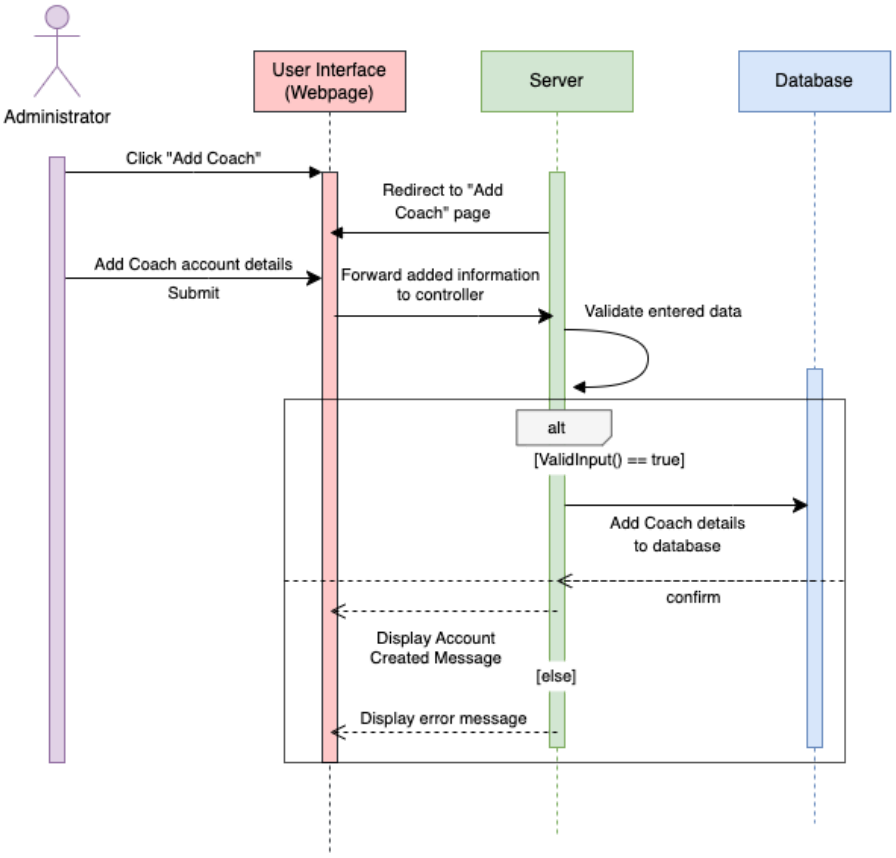


Figure 6: Sequence Diagram: Add Coach

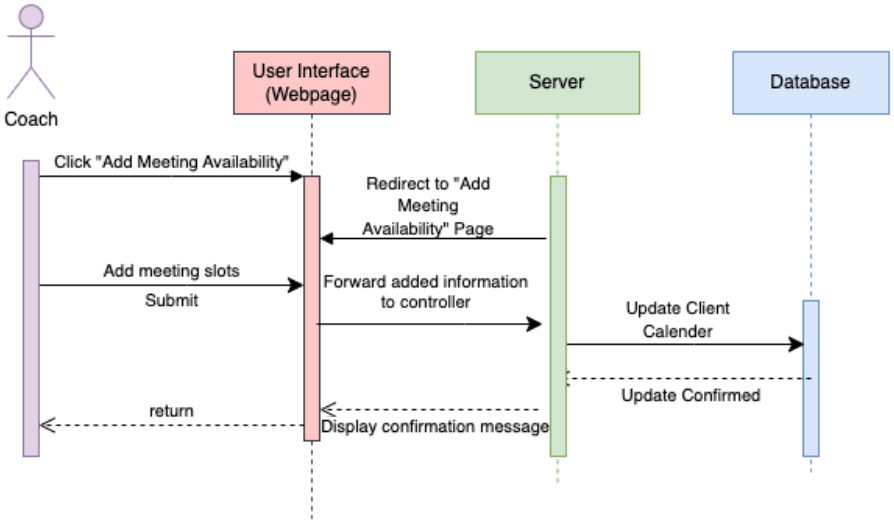


Figure 7: Sequence Diagram: Add Meeting Availability

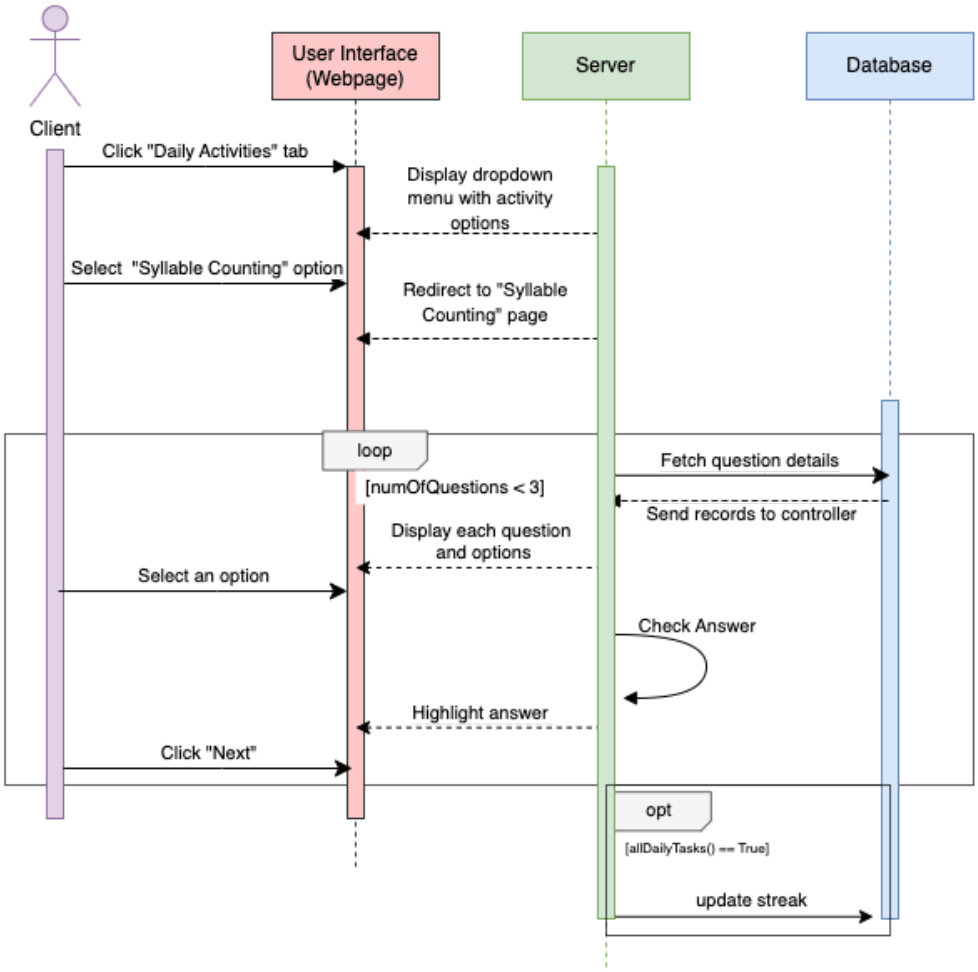


Figure 8: Sequence Diagram: Syllable Counting Activity

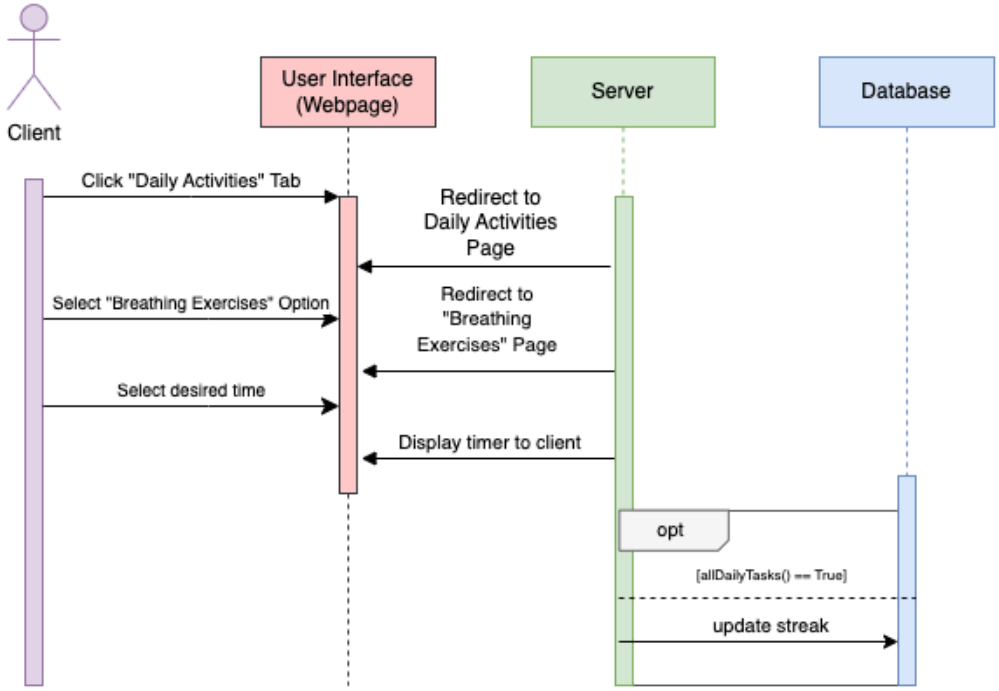


Figure 9: Sequence Diagram: Breathing Exercises

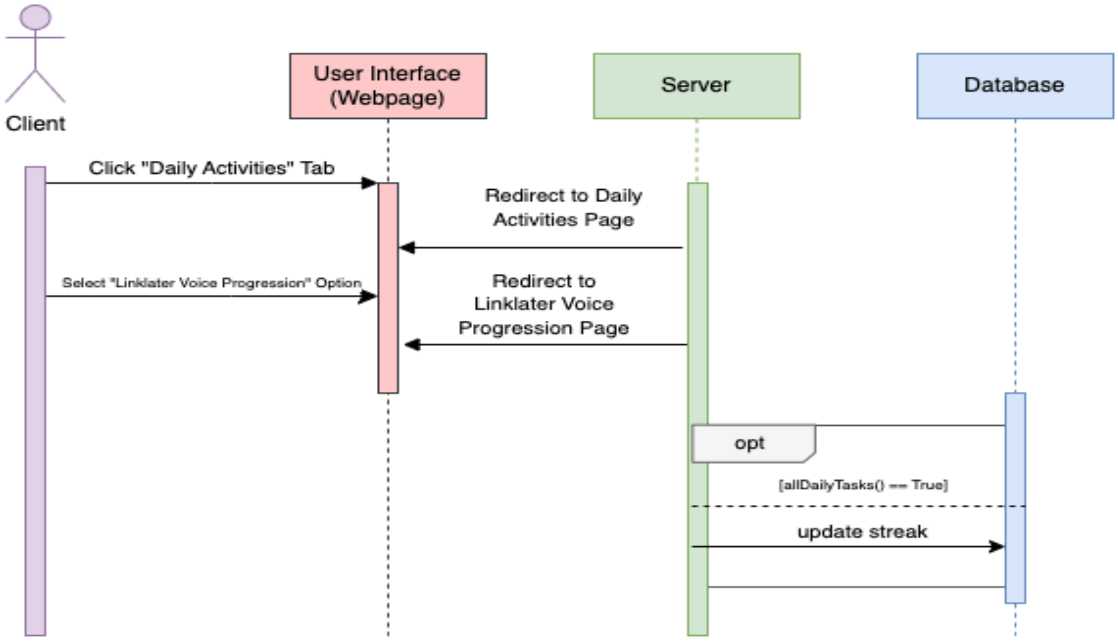


Figure 10: Sequence Diagram: Linklater Voice Progression

Next we present flow diagrams for our top three use cases, focussing more on the user interface side and where the user is redirected to on each step till the desired request is completed.

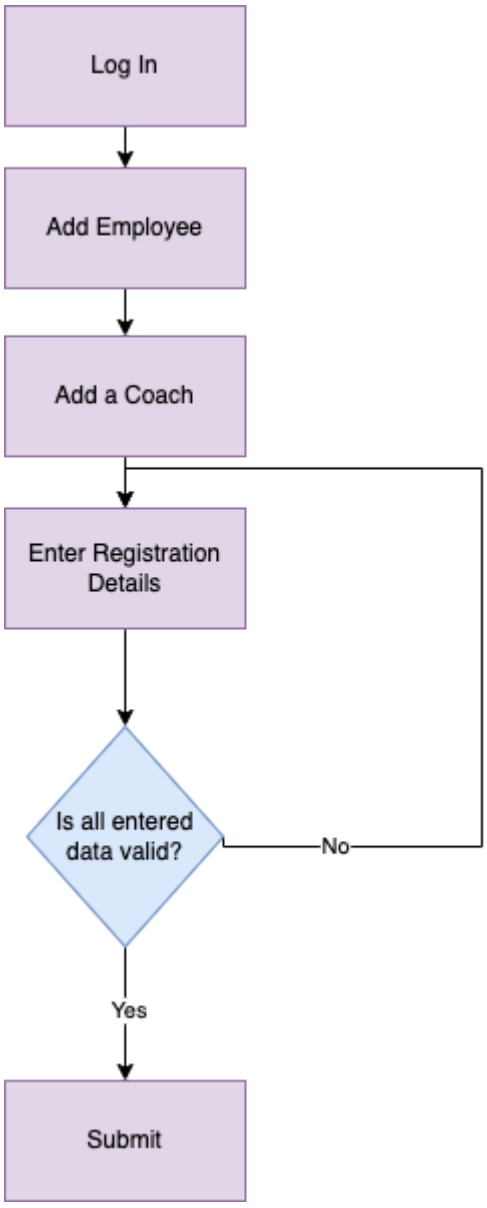


Figure 11: Flow Diagram: Add Coach

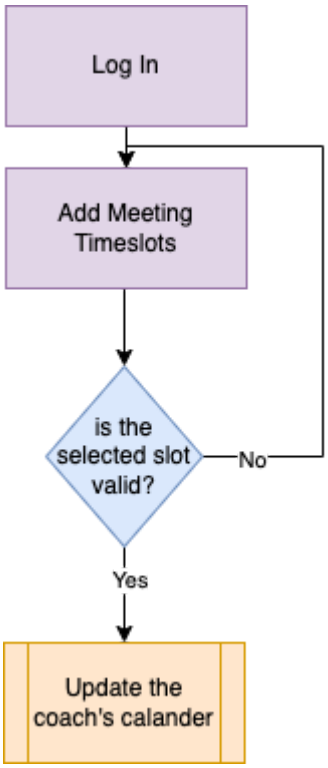


Figure 12: Flow Diagram: Add Meeting Availability

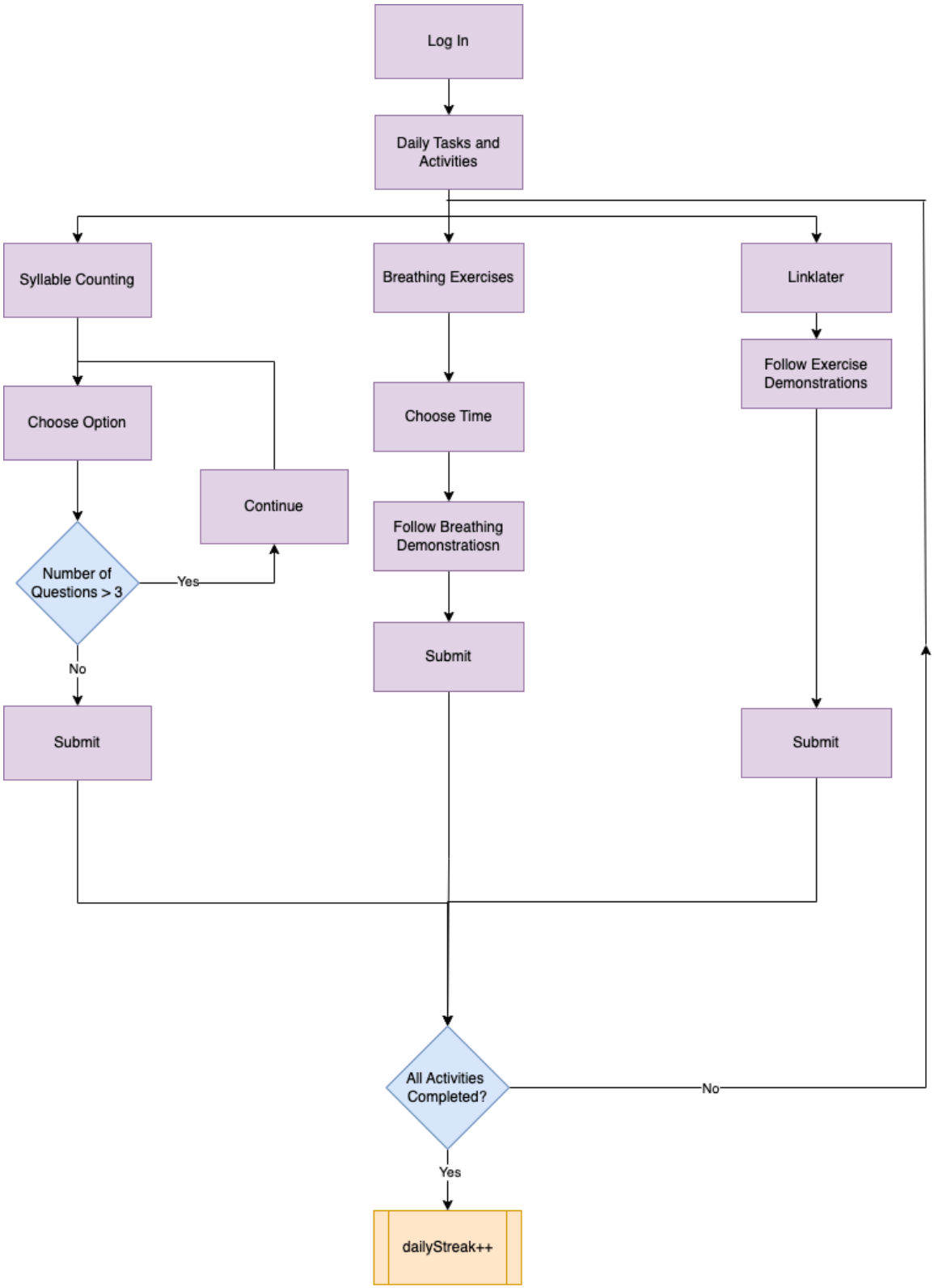


Figure 13: Flow Diagram: Daily Activities

To elaborate on our architecture further and present a deeper view of the internal interactions between the three components of our system (User Interface, Server and Database), the following diagrams will lay out the different components and functions in our implementation to highlight how the different interfaces communicate.

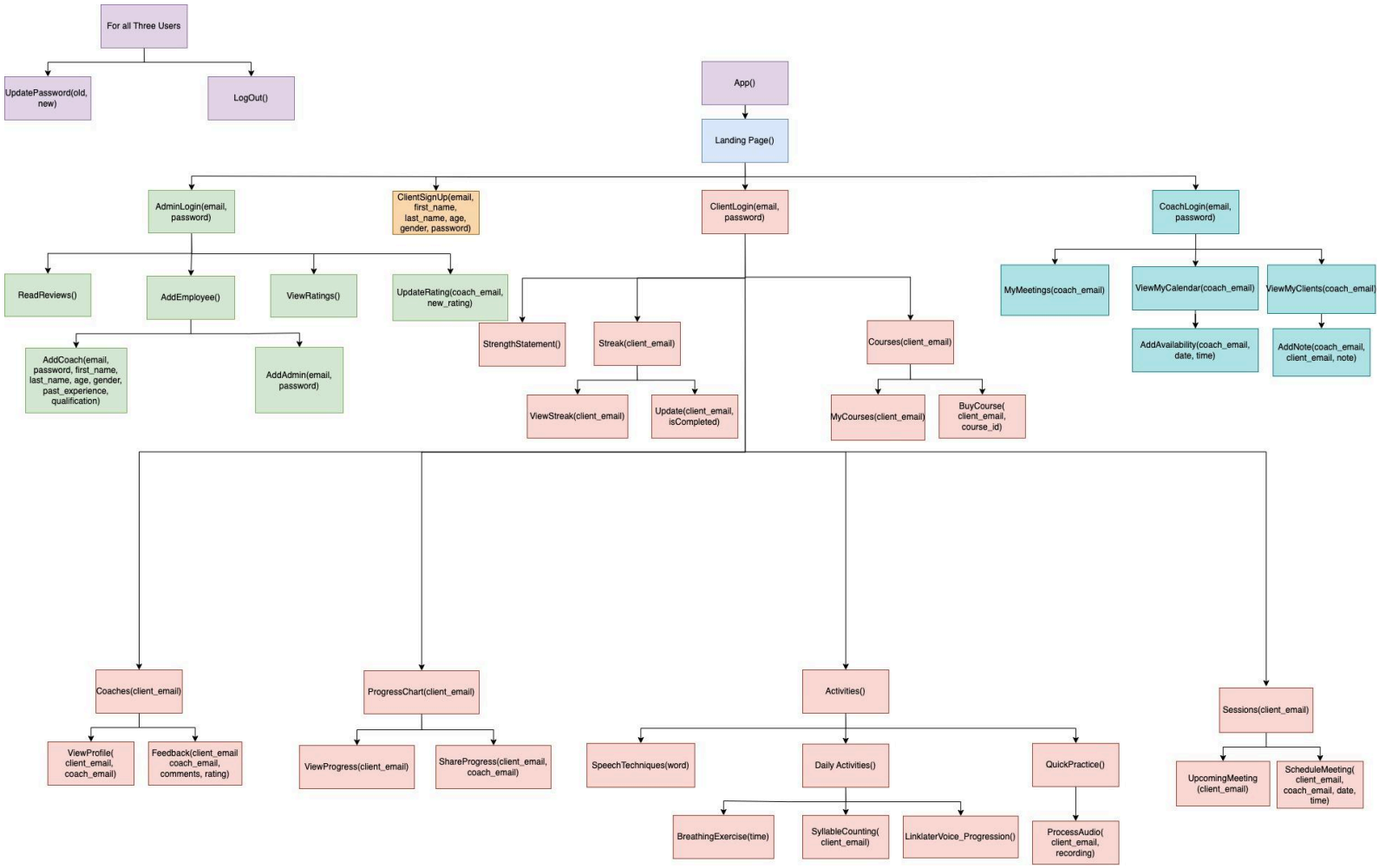


Figure 14: React Component Diagram

4.3 Data Structure

4.3.1 Internal software data structure

A speech therapy software system is composed of various components that work together to provide a comprehensive solution to speech therapy. In order for these components to communicate with each other and exchange information, they must use internal data structures. These data structures are used to pass information between the various components of the software system.

The internal data structures used in a speech therapy software system are usually designed to be simple, efficient, and flexible. They are often modeled after the data structures used in programming languages.

Some common internal data structures used in our software system include:

Arrays: Arrays are a collection of elements of the same type that are stored in a contiguous memory location. They are used to store a sequence of data such as speech sounds, words, or sentences. Arrays are often used because they provide fast access to individual elements and are easy to implement.

JSON Objects: JSON (JavaScript Object Notation) is a popular data interchange format that has gained widespread adoption in modern web applications. It is a lightweight, key-value based format that is easy to read and write, making it an ideal choice for use in web applications. Using JSON objects as an internal data structure in a speech therapy web application has several benefits. First, JSON is a lightweight format that is easy to read and write, which makes it ideal for storing and manipulating data in web apps. Second, JSON is a flexible format that can be easily modified and extended as the application evolves over time. We could use these objects as interim data structures to store the following information:

- **User profiles**: Each user could have a JSON object that stores their personal information.
- **Activities**: Each activity could be represented by an object that includes information such as the exercise name, difficulty level, and instructions.
- **Progress tracking**: The user's progress could be tracked using a JSON object that includes information such as the streak count, the exercises completed, etc.
- **Log Record Tracking**: Guftaar has a specific logging activity for clients where clients get to share their current level of stuttering that they've been facing that day.

4.3.2 Global data structure

One of the key aspects of Guftaar is the presence of global data structures that are available across different portions of the architecture.

User Profiles: User profiles are one of the most important data structures of our web application. They are used to store information about the users of the application, including their personal details, medical history, and therapy progress. User profiles are generally stored in a database and are accessible to various modules of the application.

Speech Recognition Models: Speech recognition models are another important data structure in a Speech Therapy Web Application. These models are used to recognize and analyze the speech of the users. They are usually trained on a large dataset of speech samples and can predict the words spoken by the user with a high degree of accuracy. These models are generally stored on a server and are accessible to the speech recognition module of the application.

Therapy Exercises: They are an essential part of Guftaar since they're used to help users improve their speech and communication skills. Therapy exercises are generally stored in a database and are accessible to various modules of the application, including the user interface and the speech recognition module.

Audio Files: Audio files are used to store the speech samples of the users. They are generally recorded during therapy sessions and are used to track the progress of the user. Audio files are usually stored on a server and are accessible to various modules of the application.

4.3.3 Temporary data structure

Audio Recordings: Audio recordings are one of the primary data structures used in a speech therapy software system. These recordings are created during the therapy sessions to capture the speech of the patient. The recordings are then used for analysis and evaluation by the therapist.

Speech Analysis Files: Speech analysis files are another type of temporary data structure used in a speech therapy software system. These files are created by the software to analyze the speech of the patient. The files contain information such as pitch, volume, and tone, which are used by the therapist to assess the patient's progress.

Progress Reports: Progress reports are temporary data structures used to track the patient's progress during the therapy sessions. These reports are created by the software and contain information such as the patient's speech improvement, areas that need improvement, and the next steps in the therapy process.

Session Notes: Session notes are another type of temporary data structure used to record the details of each therapy session. These notes are created by the therapist and contain information such as the patient's speech goals, the exercises performed, and the patient's response.

4.4 Database Model

4.4.1 Database schema and detailed description

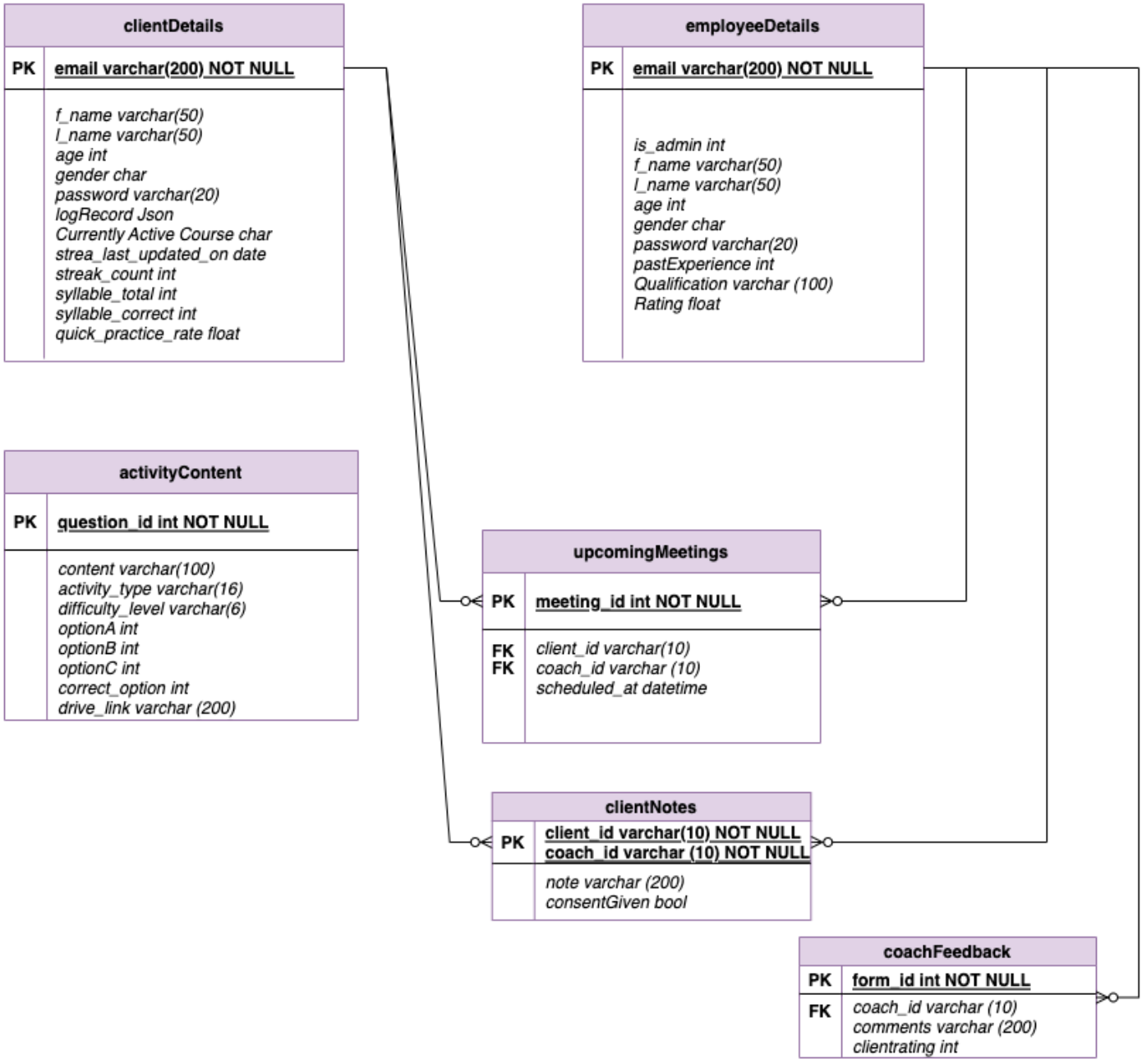


Figure 15: Database Schema

Table Name: clientDetails
Client Details is a table used to store the details of a single client registered on our web application. It has a single primary key named userName which stores the username or the user identifier of a client. It comprises 13 data members or variables whose description is provided in the following table:

Data Member/ Variable	Description
userName	This represents the username of a single client. It's also the primary key of this table. The email taken as input during sign up is parsed and the username is extracted from it.
f_name	First Name of the client taken as input during sign up.
l_name	Last Name of the client taken as input during sign up.
age	Age of the client taken as input during sign up.
gender	Gender of the client taken as input during sign up.
email	Email of the client taken as input during sign up.
password	Password of the client's account taken as input during sign up.
logRecord	This represents the current mood status of the client. It takes three valid values: No stuttering, Mild, and Extreme.
CurrentlyActiveCourse	This is the course in which a particular client is currently enrolled in.
streak_last_updated_on	This is the date on which the streak count of a particular client was last updated on.
streak_count	This is the streak count displayed on the client's dashboard.
syllable_total	These are the total syllables given as input by the client in the syllable counting activity.

syllable_correct	This is the correct answer supposed to be entered during the syllable counting activity.
quick_practice_rate	Automated rating given to the client in response to the audio recording submitted by the client in the quick practice activity.

Table 3: clientDetails Table Descriptions

Table Name: employeeDetails

Our web application represents two types of employees:

1. Administrators
2. Coaches

The table employeeDetails represents the consolidated information of the two employees. To distinguish between the employees, we have an integer flag named isAdmin that takes either a 0 or a 1. If it's a 0, it means that the employee is a coach and if it is a 1, it's an administrator. The following table provides a high-level description of the table and its associated data members.

Data Member	Description
email	This represents the email of an employee. It's also the primary key of our table.
is_admin	This is a boolean flag that signifies whether a given employee is an admin or a speech coach.
f_name	First name of an employee taken as input during registration
l_name	Last name of an employee taken as input during registration
age	Age of the employee taken as input during registration
gender	Gender of the employee taken as input during registration.
password	Password of the employee's account taken as input during registration

pastExperience	Number of years the coach has experience with speech therapy. This will be NULL for an admin.
Qualification	Represents the maximum qualification in terms of education of a speech coach.
Rating	Represents the average rating a coach's clients has given to them.

Table 4: employeeDetails Table Descriptions

Table Name: upcomingMeetings
This table stores the information of any upcoming meetings that the clients schedule with any speech instructor. Every meeting has an associated unique meeting identifier which is an integer and is also the primary key of this table. Following is a high level description of the table in the schema:

Data Member	Description
meeting_id	A unique meeting identifier of a single session a client schedules with a particular coach.
client_id	A unique identifier of the client who scheduled the meeting
coach_id	A unique identifier of the coach with whom the client scheduled the meeting with.
scheduled_at	A datetime object that represents the date and time at which the client scheduled the meeting.

Table 5: upcomingMeetings Table Descriptions

Table Name: clientNotes
This table stores all client notes that the coach logs in after the meeting with the client provided the client attends the private session on Zoom. To uniquely identify a one-to-one mapping of the client with its coach, we have both the client id and the coach id as primary keys of the table. Following is the brief description of the member variables in the table below:

Data Member	Description
client_id	A unique identifier of the client who scheduled the meeting.
coach_id	A unique identifier of the coach with whom the client scheduled the meeting.
note	Post-meeting evaluation notes typed by the coach after the meeting.
consentGiven	A boolean flag that is true if a particular client gives their consent to share their progress reports with their coach. It's false, otherwise.

Table 6: clientNotes Table Descriptions

Table Name: coachFeedback
This table represents the feedback that a client gives to a given coach on the web application. The client is not restricted to give this feedback at a particular instance but can rather visit the profile of an instructor and enter the feedback at any time. The form identifier is the primary key of this table and the form itself is an HTML-based web form that is assigned a unique identifier key. Everytime a client fills this form out, the identifier is incremented to distinguish between feedback from a given client. Following is a high level description of the table:

Data Member	Description
form_id	A unique identifier of the feedback form. It's also the primary key of the table.
coach_id	A unique identifier of the coach for whom the client has given feedback.
comments	A string representing a series of comments that a client adds in the form.
clientRating	An integer bounded between 1 and 5. 1 being not satisfied with the coach and 5 being very satisfied.

Table 7: coachFeedback Table Descriptions

Table Name: activityContent
This table represents the activities that are offered by our web application. The flow of every activity is highlighted by a range of questions where every question is uniquely identified by a unique identifier. Every question is a multiple choice question where the clients can answer the question by picking a single correct option from the three given options named A, B, and C. The question identifier is also the primary key of this table. Following is a high level description of the data members stated in the table:

Data Member	Description
question_id	A unique identifier separating every question.
content	Content of the question (the problem statement)
activity_type	A string that takes three valid literals: Syllable Counting, Quick Practice and Speech Techniques.
difficulty_level	A string that accepts three valid literals: Easy, Medium, and Hard.
optionA	Answer choice given as an option A.
optionB	Answer choice given as an option B.
optionC	Answer choice given as an option C.
correct_option	Correct answer stored with respect to the given question.
drive_link	Client's will submit their audio recording files via Google Drive for the Quick Practice activity, and they'll submit that on the web portal.

Table 8: activityContent Table Descriptions

Table Relationships

Table 1 Name	Table 2 Name	Relationship	Rationale
clientDetails	upcomingMeetings	1 to Many (Optional)	There can be multiple meetings booked by the same client, causing their ID to be referenced in multiple records of the upcomingMeetings table. Alternatively, the client may not have booked any meeting thus far, indicating that this relationship is optional.
clientDetails	clientNotes	1 to Many (optional)	Given that there can be multiple meetings booked by the same client, under the circumstance that the coach adds a note for that client, their ID may be referenced in multiple records of the clientNotes table. Alternatively, the client may not have booked any meeting thus far, or the coach may choose to not add any notes post meeting, indicating that this relationship is optional.
employeeDetails	upcomingMeetings	1 to Many (Optional)	Employees may have multiple meeting slots booked by clients, and so can appear in multiple records of the upcomingMeetings table. It may also be the case that an employee does not have any meeting slots booked, thus making this relationship optional.
employeeDetails	clientNotes	1 to Many (optional)	Given that there can be multiple meetings booked with the same coach, their ID may be referenced in multiple records of the clientNotes table if they choose to add notes. Alternatively, it

			is possible that an employee does not have any meeting slots booked, and so, has no clients, thus making this relationship optional.
employeeDetails	coachFeedback	1 to Many (optional)	Given that multiple clients are able to book meetings with coaches, there can be many reviews submitted for the same coach, causing the coachID to be repeated across records in the coachFeedback table, incurring a one to many relationship. It is possible that either no client has interacted with a specific coach, or chooses to not share any feedback, in which case this relationship is optional.

Table 9: Table Relationships

4.4.2 Database

MongoDB Atlas is a highly flexible document-based database that offers a more versatile approach to data modeling in comparison to traditional relational databases like MySQL. With MongoDB Atlas, data can be easily stored and accessed in a variety of formats, without having to conform to strict schemas. This is further enhanced by its enriched API, which is designed for multiple backend programming languages, saving the hassle of writing multiple SQL queries manually. As a web application scales, SQL queries become increasingly complex, often becoming nested and difficult to interpret. In contrast, MongoDB Atlas is a fully-managed cloud-native database service, eliminating the need to manage infrastructure or set up backups and disaster recovery. MySQL, however, is limited in its ability to handle congestion and high data payloads. Its services are deployed in designated data centers, making it less efficient when managing concurrent users.

4.5 External Interface Requirements

4.5.1 User Interfaces

Given that the primary aim and purpose of our project is to provide a platform where those looking to improve their speech articulation can self-reliantly access certified speech therapy services, a graphical user interface is a primary requirement for the ease of our users. Therefore, our webportal will be built on ReactJS (a framework of JavaScript), and will support GUI.

We will have a standard navigation bar at the top with all the navigation links to the main features of our web application (displayed in Section 5). Whenever the user is required to input commands and information, our system will ensure to give them feedback in a timely manner upon incorrect commands. In addition, the application will be able to handle any errors and provide helpful instructions for resolving the issue. Our design goal is to aid all our user functionalities with interactive icons, graphics and multimedia content (audio and video support) to make speech therapy a seamless process with Guftaar.

4.5.2 Hardware Interfaces

The web application will work on all computing devices including mobile smartphones that have a web browser installed capable of running and rendering JavaScript code. Moreover, for smooth loading of the web page, it is advisable to access the portal on devices that have more than 2GB RAM. In addition, like every other website, our application requires a stable internet connection which necessitates the compulsion of having the relevant hardware components such as routers, switches and modems. A steady internet connection and such storage requirements are especially needed to support the graphical user interface requirements specified in section 3.2.1. To seamlessly interact with the services provided by our platform, a functioning microphone alongside a keyboard and mouse will also be necessary.

5 User Interface Design

5.1 Description of the user interface

ReactJS:
The main frontend framework that Guftaar utilizes is ReactJS. ReactJS has been chosen over other frontend frameworks due to the following reasons:

- Modularity: React's architecture allows bundling frontend code into custom components, thus providing easier management of complex and large systems.
- Integration with Third Party APIs: Embedding react components within other technologies and APIs is a simple process. Given Guftaar's reliance on APIs such as Calendly, easy integration with third-party APIs became an important deciding factor for frontend frameworks.
- Fast Rendering: React's advantage over other frameworks is that it creates a virtual DOM as a copy of the real DOM. This allows React to do minimum re-rendering upon a change in the DOM. This translates to faster rendering and better performance.
- Strong Community Support: ReactJS is the number 5 top-rated repository with over 160k stars and over 5 million applications publicly admitting to using it [2]. With web apps where the systems tend to grow into large codebases, it is very useful for developers to have strong community support to navigate the complexity of the app.

Tailwind CSS:
For bringing Guftaar's Figma screens to reality, the styling library that will be used is Tailwind CSS. The main reasons to choose Tailwind as the styling library are the following:

- Small file size: Large custom CSS files can result in poor page loading times. Guftaar's responsiveness guarantees the use of Tailwind, which uses Brotli compression and PurgeCSS to reduce file sizes [3].
- Easy customization: One of the main reasons for Guftaar to pick Tailwind over other libraries such as Bootstrap is the great range of customizations and the ability to override default configurations.

Figma:
Figma is a collaborative web application for interface design. The feature set of figma focuses on user interface and user experience design, utilizing a variety of vector graphics editors and prototyping tools. Figma is used to make the screens for Guftaar's 3 most important use cases.

MVC Presentation Layer:
The presentation layer contains all the user interfaces of an application. It contains MVC controllers, views, static files, JavaScript files, and other client side frameworks. The view component comprises the UI for Guftaar which the users will interact with. The controller consists of application code which interacts with the view and the model.

5.2 Information architecture

The landing page for Guftaar allows the user to select their role (Coach, Admin, Client). Upon selecting the role, they can log in to the Guftaar platform. Clients are also allowed to sign up if they are not already registered on the Guftaar platform. Once the user is logged in, they are led to their respective dashboards. Each dashboard represents the user's functionality. All flows are depicted in the information architecture diagrams below. For the sake of simplicity, the diagrams for all actors have been separated.

Client's Information Architecture:

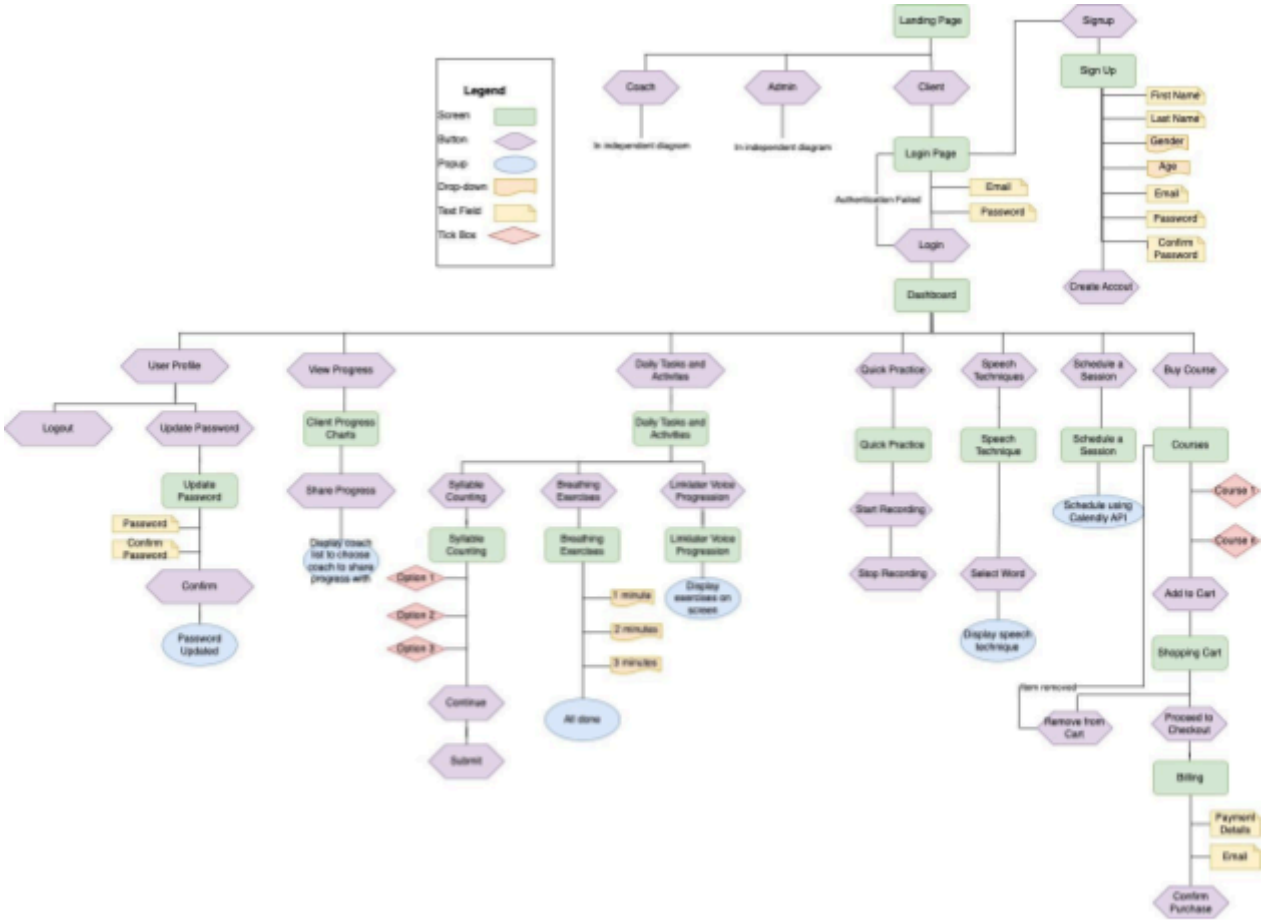


Figure 16: Client Information Architecture Diagram

Coach's Information Architecture:

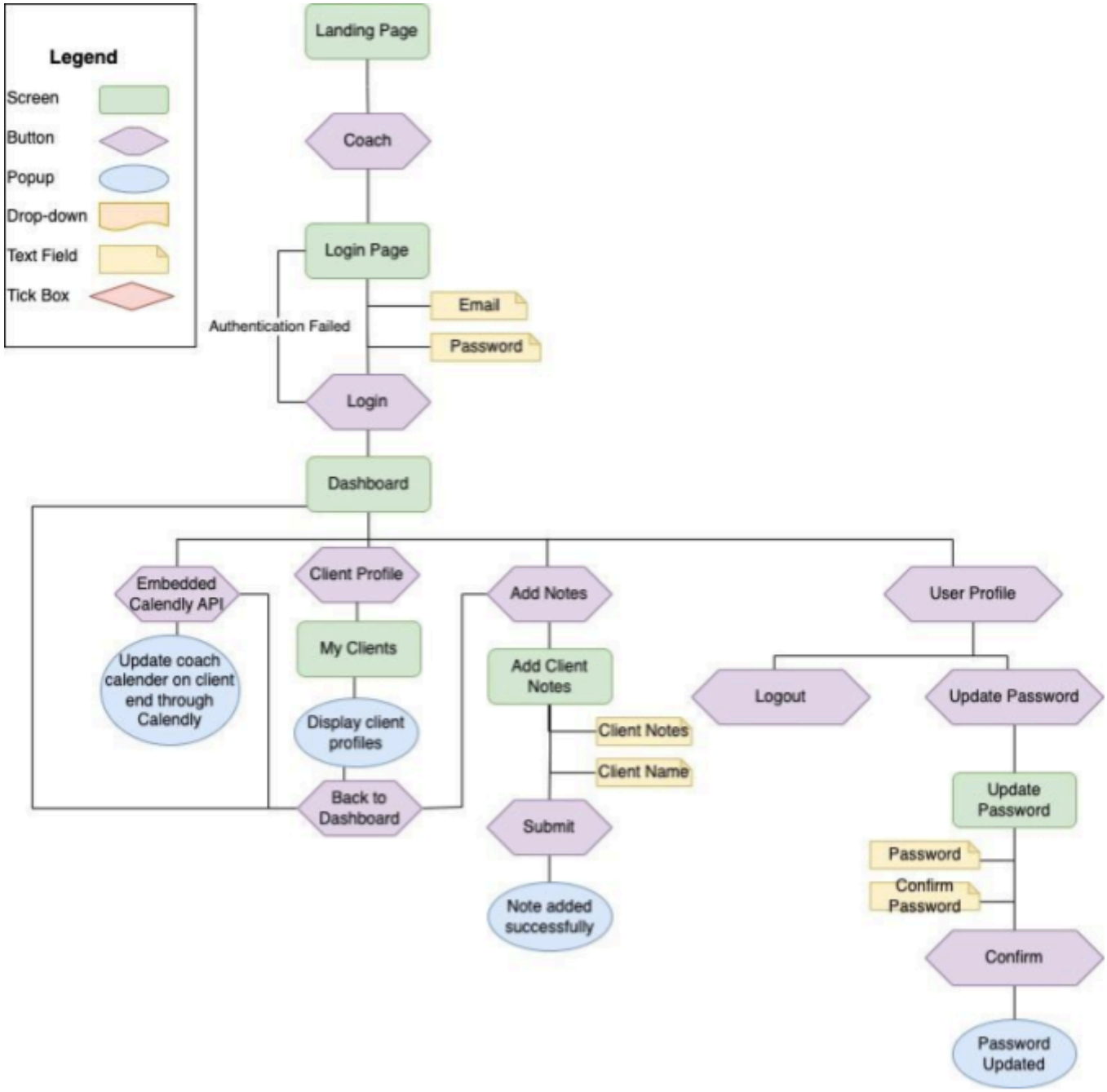


Figure 17: Coach Information Architecture Diagram

Admin's Information Architecture:

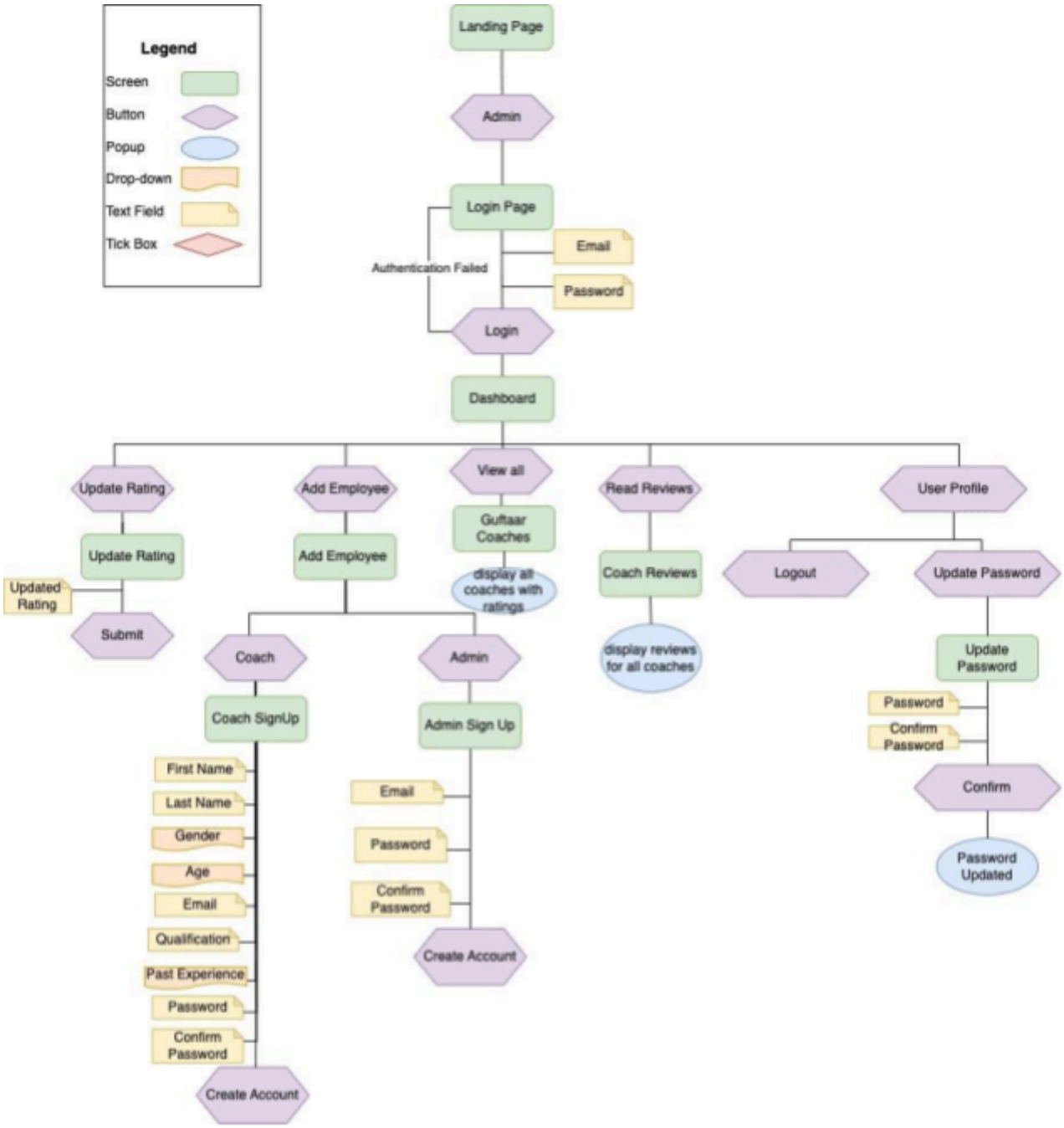


Figure 18: Admin Information Architecture Diagram

5.3 Screens

The following screens cover the top use cases of each actor, and illustrate the flow that the user will undertake to be able to perform the required action.

All screens have been designed on Figma, and are mapped to the standard 16:9 size in order to maintain the most generality. All Material Design guidelines for Web, in case of icon choice, font sizes, and components have been followed in order to ensure direct translation to the development version using the React MUI library.

Each screen is explained with its motivation, design elements and corresponding choices, as well as use cases covered below:

1. **Landing Page:**

Figure 19: Landing Page

Goal:
Figure 5.3.1 represents the single page view of the landing screen for the Guftaar web application. This screen introduces users to the platform, and provides options for PWS to sign up, and coaches as well as administrators to log in.

It provides a high level overview of the speech support services the platform provides, and contains a call-to-action, encouraging prospective users to sign up and avail Guftaar services.

User Interface:

The UI for Figure 19 consists of the following elements:

Landing Page	
Button	**Functionality**
"I'm looking for speech support"	Navigate the client to the login/sign-up page for PWS.
"I'm a coach that offers services"	Navigate the coach to the log-in page.
Admin Login	Navigate the admin to the administrator log-in page.
Sign Up Now	Navigate the client to the login/sign-up page for PWS.

Table 10: UI Components for Figure 19

The user interface for the landing page has been intentionally designed to grasp user attention. The use of large readable text, supporting graphics and appropriate color blocking will make it easier for users to navigate through the page. The use of shadow/hover effects indicates that the sign-up/login buttons are easily clickable, and can be perceived as such as well. The page intentionally has minimal text and buttons so as to avoid overwhelming first time visitors.

The use of one main purple color is meant to be a source of comfort for users, and build an association of cool tones normally used for themes like health.

Use Case: The landing page serves as the webpage that enables users to log-in as well as create an account on the Guftaar platform, in the case of clients. This serves as the gateway for users to log in to the web application, and avail services depending on the type of actor (e.g. client, coach or administrator).

2. Client Dashboard:

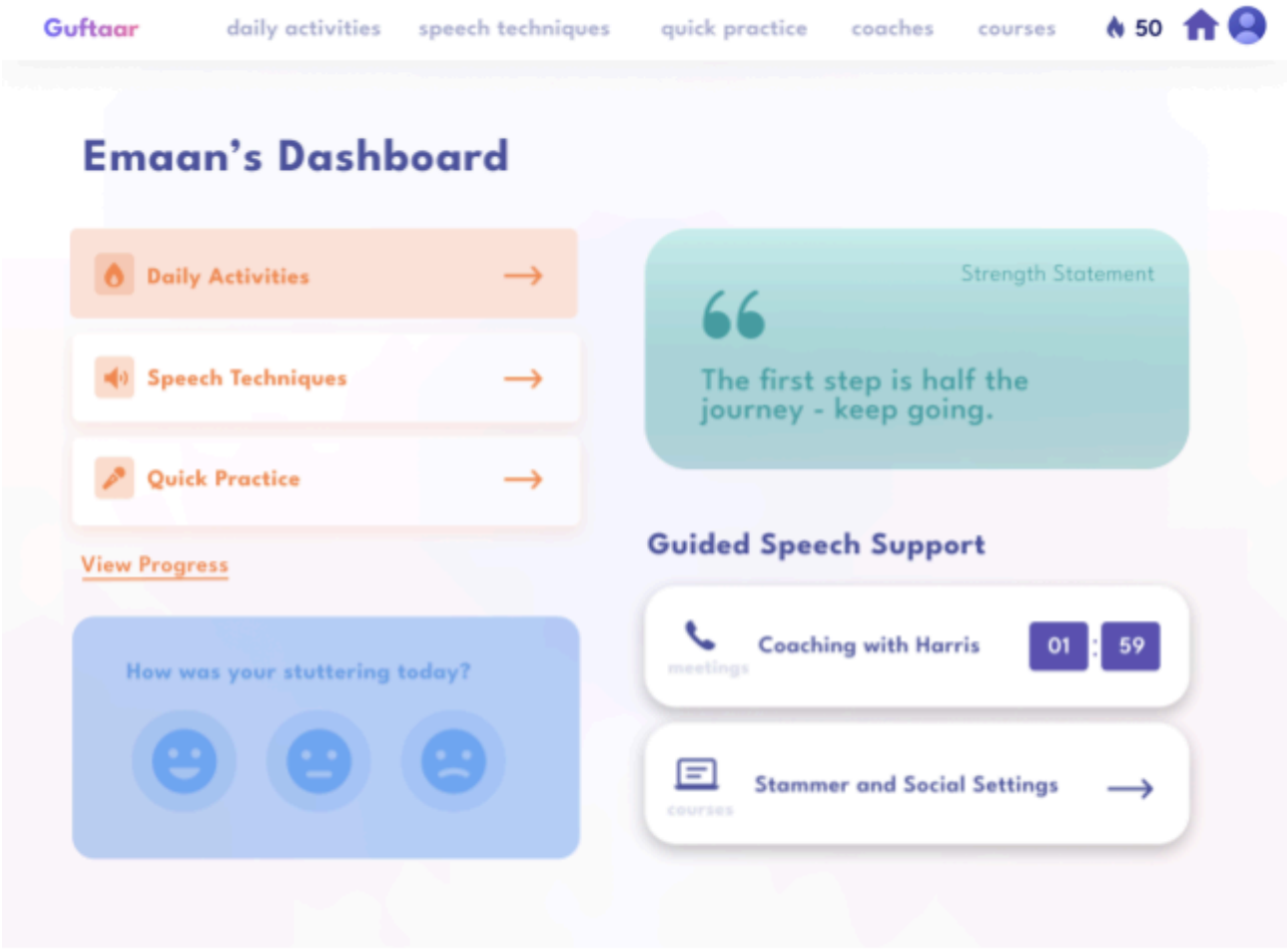


Figure 20: Client Dashboard

Goal: Figure 20 represents the client dashboard. The dashboard appears once the client has successfully logged into the Guftaar platform, and it encapsulates the major functionality available to the client. The goal of the dashboard is to provide easy accessibility to major functionality. Clients can navigate to daily tasks and activities, speech techniques, and quick practice. They can also schedule a session or buy a course through the dashboard. Moreover, the dashboard displays strength statements to keep the clients motivated throughout their speech therapy journey. Lastly, clients can also report their daily stuttering through the dashboard. The navbar assists the dashboard in providing accessible routing to aforementioned functionality, and additionally also lets clients view coach profiles, client profile, and current streak score, as well as their progress so far which visualizes their activity on the platform,

User Interface: The UI for Figure 20 consists of the following elements:

Navbar	
Button	**Functionality**
Daily Activities	Navigate the client to the daily activities page.
Speech Techniques	Navigate the client to the speech techniques page.
Quick Practice	Navigate the client to the quick practice page.
Coaches	Navigate the client to the coaches page.
Courses	Navigate the client to the courses page.
Home	Navigate the client to the dashboard.
Client Profile	Display a popup where client can either logout, or update password

Table 11: UI Components for Navbar

Dashboard	
Button	**Functionality**
Daily Activities	Navigate the client to the daily activities page.
Speech Techniques	Navigate the client to the speech techniques page.
Quick Practice	Navigate the client to the quick practice page.
Coaches	Navigate the client to the coaches page.
Meetings	Display countdown till upcoming meeting with a coach.
Daily Log	Record client's daily stuttering response.
View Progress	Navigate the client to a page visualizing progress like their streak, task performance, and mood

Table 12: UI Components for Dashboard

The UI for the dashboard has been designed with user-friendliness and ease-of-use being primary considerations. The UI for the dashboard utilizes compartmentalization and a grid-like layout to ensure all functionally is organized in a clutter-free fashion. To enhance usability, user interface

components support drop-shadows and hover-effects. Guftaar also strives to make the platform accessible to all. In that regard, Guftaar utilizes an easy-to-read font family (Sans-serif) and a color-blind neutral color palette. Moreover, Guftaar uses graphics wherever possible to increase the user-friendliness of the platform.

Use Case: The dashboard serves as an amalgamation of major client use cases. It is a portal through which major functionality is made available to the client within a single click. It is organized in such a way that the use cases are easy to follow.

3. **Daily Tasks and Activities:**

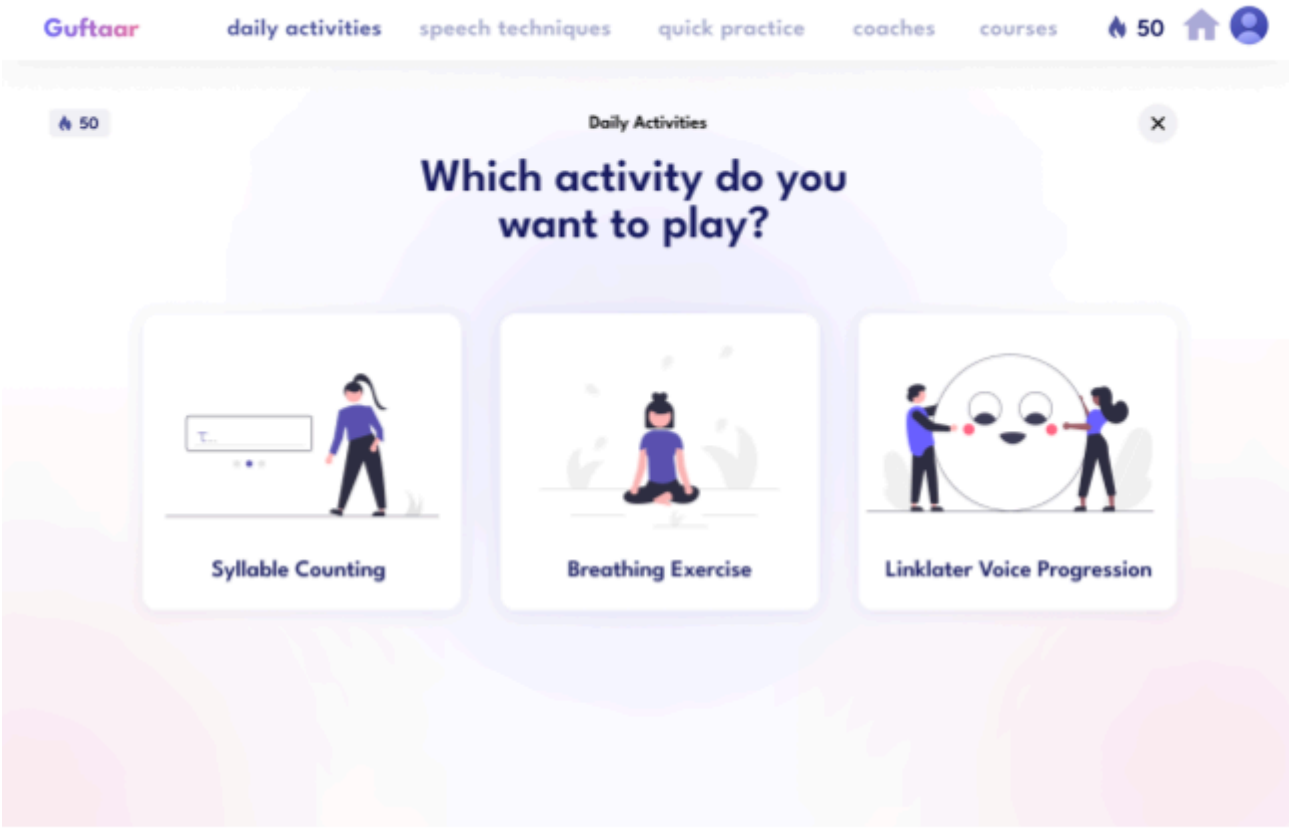


Figure 21: Daily Activities

Goal: Figure 21 represents the Daily Tasks and Activities Page. The screen allows clients to access 3 different daily activities through their corresponding buttons: syllable counting, breathing exercise, and Linklater voice progression. The screen also displays the on-going streak of daily activities for

the client. The goal of the screen is to serve as a portal for individual components within the daily tasks and activities.

User Interface: The UI for Figure 21 consists of the following elements:

Button	Functionality
Syllable Counting	Route the client to the syllable counting page (screen shown below)
Breathing Exercise	Route the client to the breathing exercise page (screen shown below)
Linklater Voice Progression	Route the client to the Linklater voice progression page (screen shown below)
Exit	Route the user to the client dashboard

Table 13: UI Components for Figure 21

To stay consistent with the client's mental models and ease of use, UI components (especially the main functionality), have been placed in a centralized location on the page. For better usability and user experience, large buttons have been utilized with drop shadows and hover effects (not shown), to indicate that the UI component can be clicked. Graphics and signifiers have been included to make the screen more user-friendly. Moreover, a large and readable (sans-serif) font family is used to enhance the accessibility of the Guftaar platform. Lastly, it has been ensured that the theme, font, layout, and color palette is consistent with the rest of the application.

Use Case: Figure 21 depicts the main use case for clients. As mentioned in the SRS, clients can complete three different types of daily activities (separate screens for each activity are shown below). Based on successful completion, the streak value for the client is updated.

4. Syllable Counting Exercise:

Figure 22 (a): Syllable Counting Activity

Figure 22 (b): Syllable Counting Activity

Goal: Figure 22 represents the first activity within the Daily Activities page. This screen provides the entire functionality of syllable counting, where the screen displays 3 questions and corresponding options. Based on the option selected, the screen displays if the answer entered is correct or not (the correct case is shown in Figure 22 (b)).

User Interface: The UI for Figure 22 consists of the following elements:

Button	Functionality
Option A	Record the client response as option A.
Option B	Record the client response as option B.
Option C	Record the client response as option C.
Continue	Display the next syllable counting question.
Exit	Redirect to the daily activities page.

Table 14: UI Components for Figure 22

The UI for syllable counting ensures that it is consistent with the rest of the Guftaar web app. The principles of accessibility, user-friendliness, usability, and ease of use extend to this screen as well. To stay consistent with mental models for option/choice based questions, the options have been displayed in the center of the screen and have been properly labeled. Alongside the usability features such as shadows and hover-effect, the Guftaar app also displays if the option selected is correct or not in a way which stays consistent with the client's mental models. To further enhance user experience, Guftaar provides an additional information/help button in case a client finds the web-page hard to navigate.

Use Case: Figure 22 presents the first component of the Daily Tasks and Activities use case, which is the major use case for clients of Guftaar. 3 syllable counting questions are displayed to the user and the response is recorded.

5. Breathing Exercise

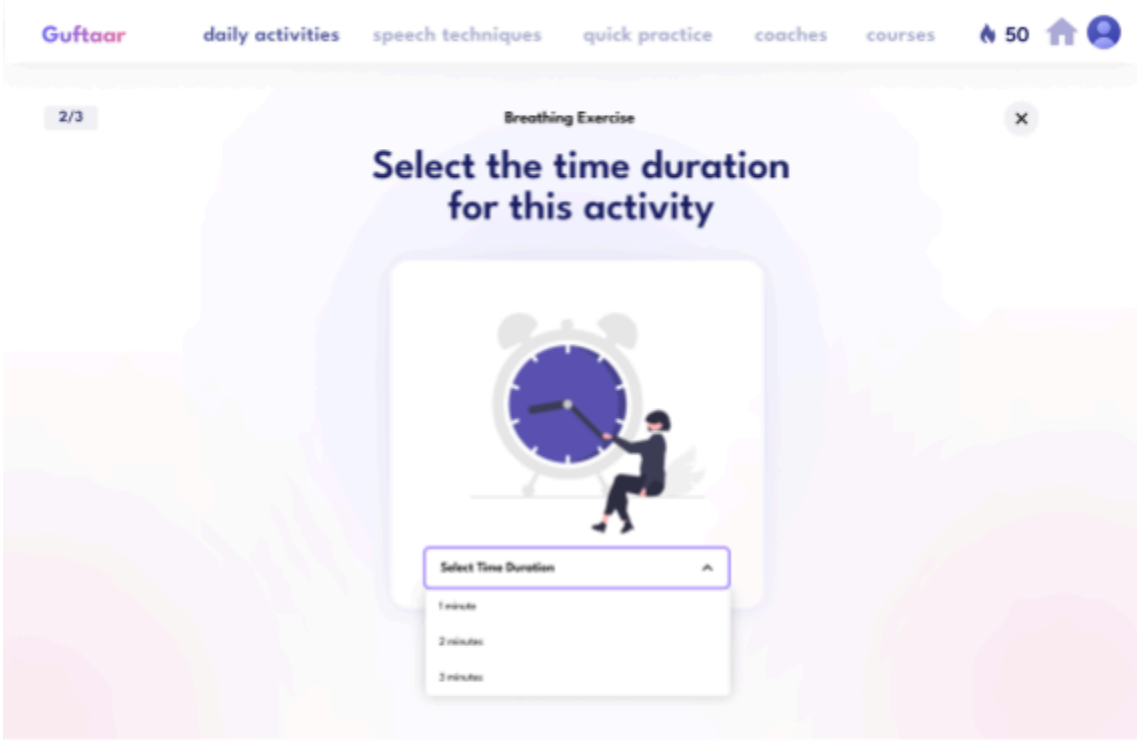


Figure 23 (a): Breathing Exercise

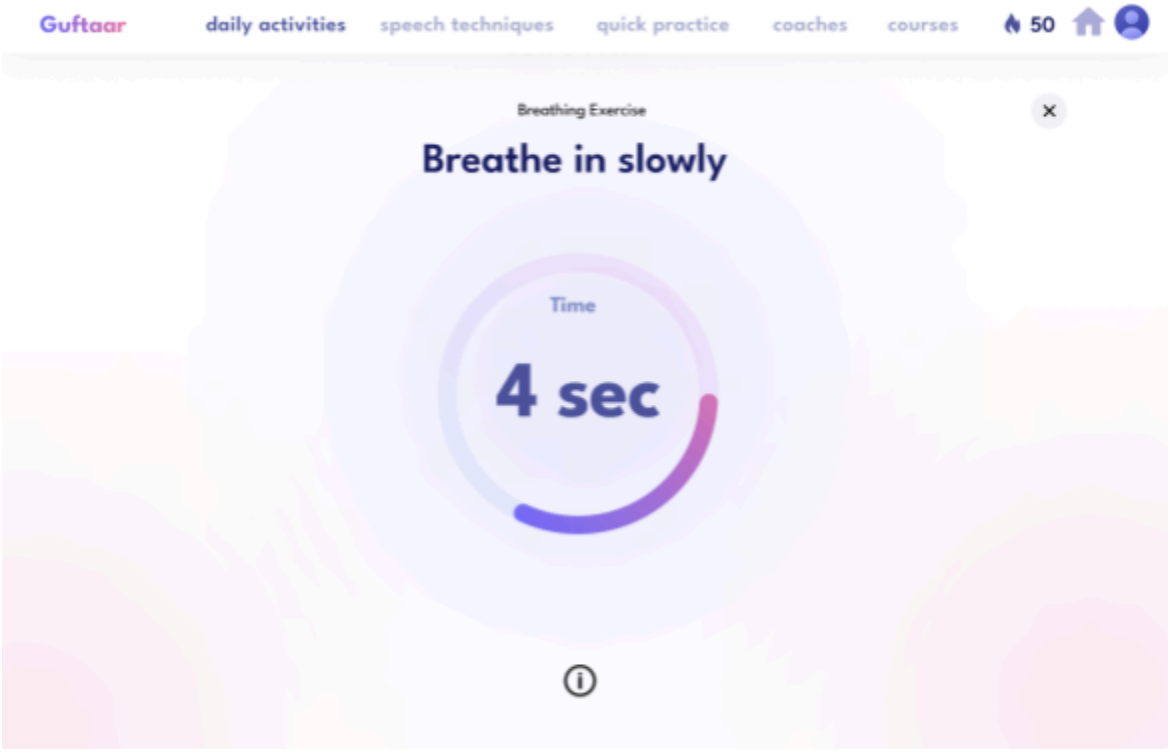


Figure 23 (b): Breathing Exercise

Goal: Figure 23 depicts the Breathing Exercise screen. This screen serves the entire functionality of the breathing exercise use case and the client can navigate to the screen through daily tasks and activities. The goal of the screen is to guide the client through the breathing exercise procedure based on the time selected through visual timers and instructions.

User Interface: The UI for Figure 23 consists of the following elements:

Button	Functionality
Select Time Duration	Record the duration for which the client wants to perform the exercise.
Exit	Redirect to the daily activities page.

Table 15: UI Components for Figure 23

The UI for breathing exercise has been designed with considerations for simplicity and visual assistance/guidance for clients. A centrally located component with relevant supporting graphics brings attention to the only source of input required from the user. Highlights and hover-effects ensure that the user is aware of the option that is selected. Once the time is selected, a visual representation of the timer, carefully crafted to ensure ease-of-use (through both numerical and circumference depiction of time), guarantees no compromise in user-experience. In case the screen is hard to follow, Guftaar also provides an additional/help button. Figure 23 also ensures that the UI is consistent with the rest of the platform.

Use Case: Figure 23 represents the second component of Daily Tasks and Activities, the most important client use case for Guftaar. Based on the input time, Guftaar assists clients in following the ‘X-X-X’ breathing strategy, where X is a chunk of time, and the client inhales, holds, and exhales for that amount of time.

6. Linklater Voice Progression

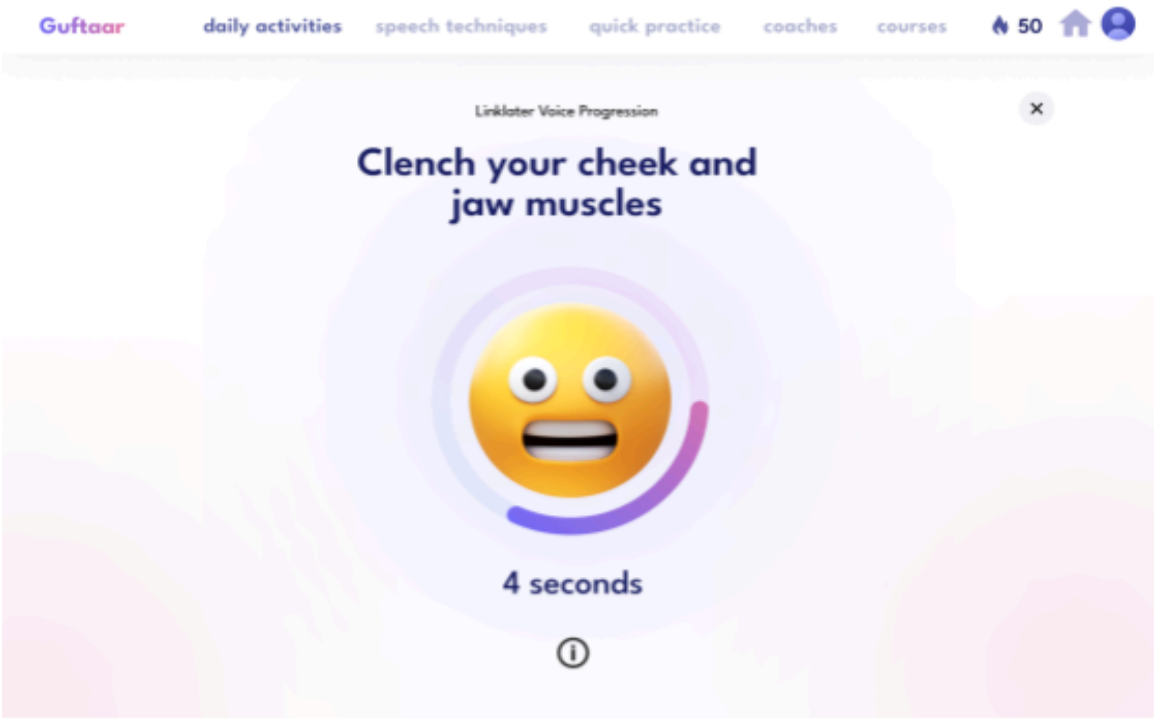


Figure 24 (a): Linklater Voice Progression

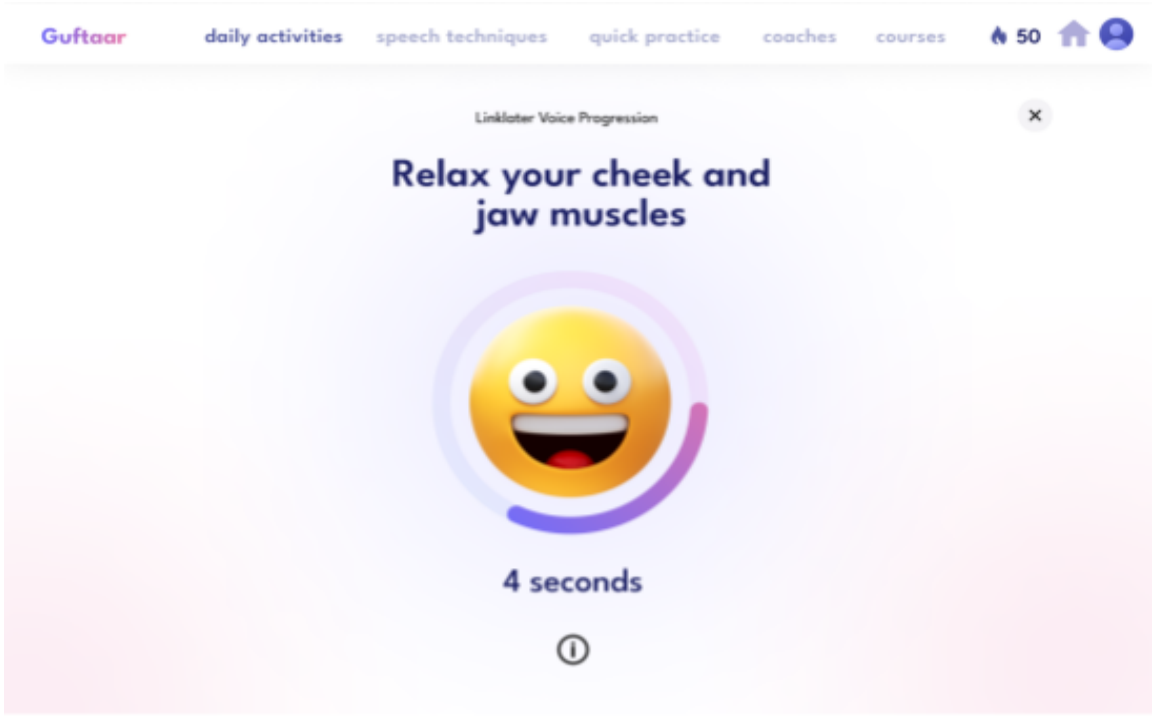


Figure 24 (b): Linklater Voice Progression

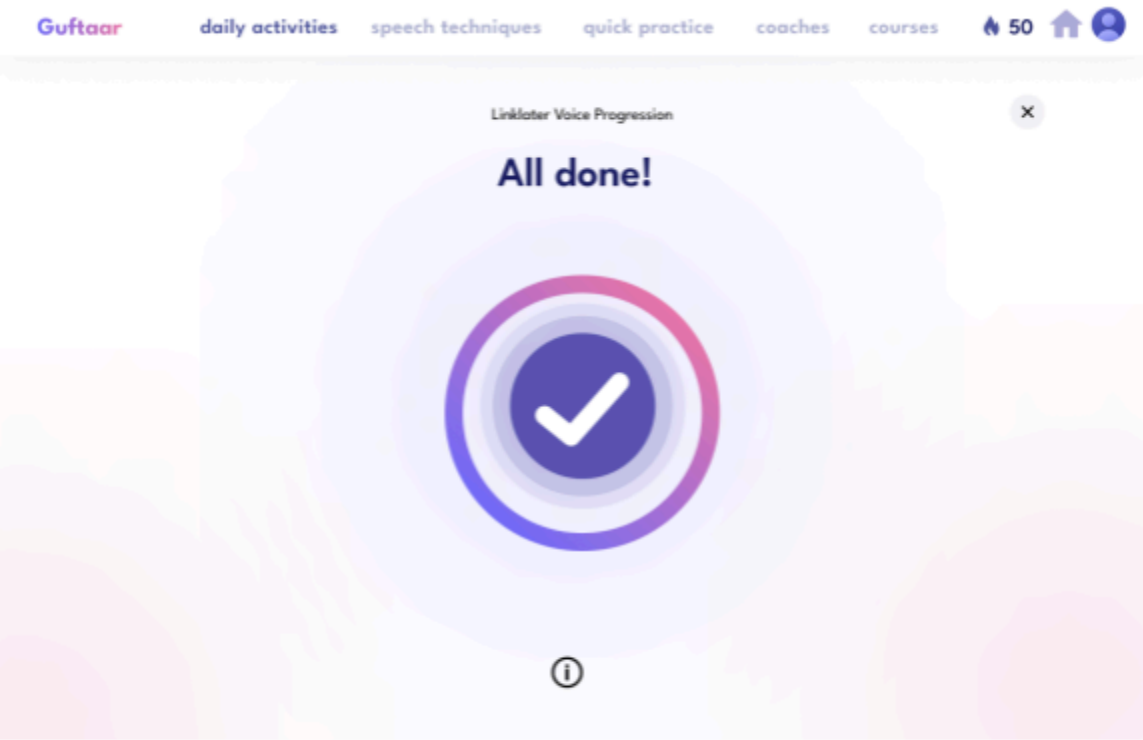


Figure 24 (c): Linklater Voice Progression

Goal: Figure 24 depicts the Linklater Voice Progression screen flow. These screens serve as the entire functionality of the voice progression exercise use case meant to exercise cheek and jaw muscles, and the client can navigate to the screen through daily tasks and activities. The goal of the screen is to guide the client through the procedure through visual timers and cues for expressions that can be emulated, and enable the necessary exercise of the muscles to be used.

User Interface: The UI for the Figure 24 consists of the following elements:

Button	Functionality
Exit	Redirect to the daily activities page.

Table 16: UI Components for Figure 24

The UI for the Linklater Voice Progression activity has been designed with considerations for simplicity and visual guidance for clients. A centrally located component with relevant supporting graphics draws attention required from the user for task completion. The use of visual expressions help clients emulate the expression needed for successful Linklator exercise of the cheek and jaw. In case the screen is hard to follow, Guftaar also provides an additional/help button. Figure 24 also ensures that the UI is consistent with the rest of the platform.

Use Case: Figure 24 represents the third and final component of Daily Tasks and Activities, the most important client use case for Guftaar. Based on the reputable Linklator method of voice training, the use case follows a step-by-step practical series of exercises that begin with clenching, followed by relaxation of the jaw muscles, known to assist with stammering.

7. Admin Dashboard

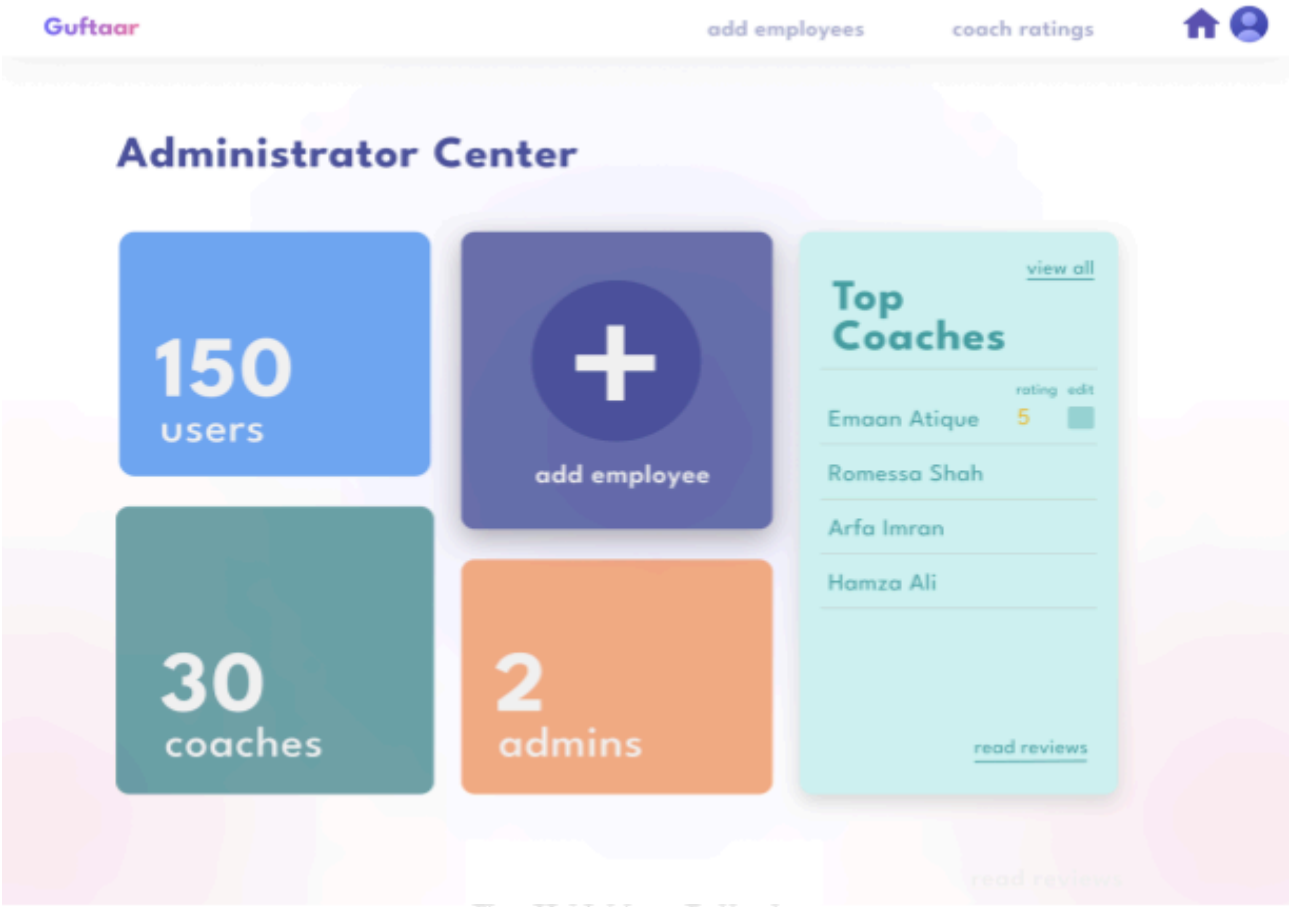


Figure 25: Administrator Dashboard

Goal: Figure 25 represents the admin dashboard. The dashboard appears once the admin has successfully logged into the Guftaar platform, and it encapsulates the major functionality available to the admin. The goal of the dashboard is to provide easy accessibility to major functionality. Admins can navigate to add employees, can view coach ratings, and can also read reviews posted by clients. This functionality is available both through dashboard and through navbar, to facilitate the admin.

User Interface: The UI for the Figure 25 consists of the following elements:

Navbar	
Button	**Functionality**
Add Employees	Navigate the admin to the add employees page.
Coach Ratings	Navigate the admin to the coach ratings page.
Home	Navigate the admin to the dashboard.
Admin Profile	Display a popup where admin can either logout, or update password

Table 17: UI Components for Navbar

Admin Dashboard	
Button	**Functionality**
Add Employees	Navigate the admin to the add employees page.
View All	Navigate the admin to the coach ratings page.
Read Reviews	Navigate the read reviews page.

Table 18: UI Components for Dashboard

The UI for the dashboard has been designed with user-friendliness and ease-of-use being primary considerations. The UI for the dashboard utilizes compartmentalization and a grid-like layout to ensure all functionally is organized in a clutter-free fashion. To enhance usability, user interface components support drop-shadows and hover-effects. Guftaar also strives to make the platform accessible to all. In that regard, Guftaar utilizes an easy-to-read font family (Sans-serif) and a color-blind neutral color palette. Moreover, Guftaar uses graphics wherever possible to increase the user-friendliness of the platform.

Use Case: The dashboard serves as an amalgamation of all admin use cases. It is a portal through which all functionality is made available to the admin with a single click. It is organized in such a way that the use cases are easy to follow.

8. Add Employees

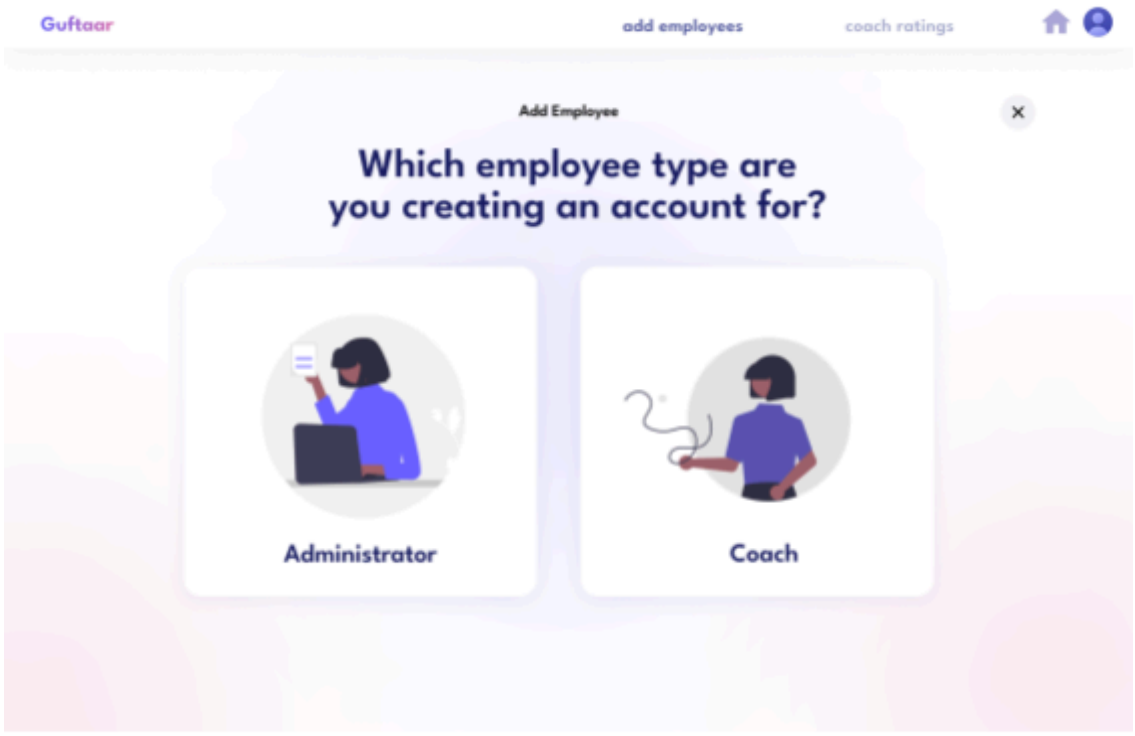


Figure 26 (a): Add Employees

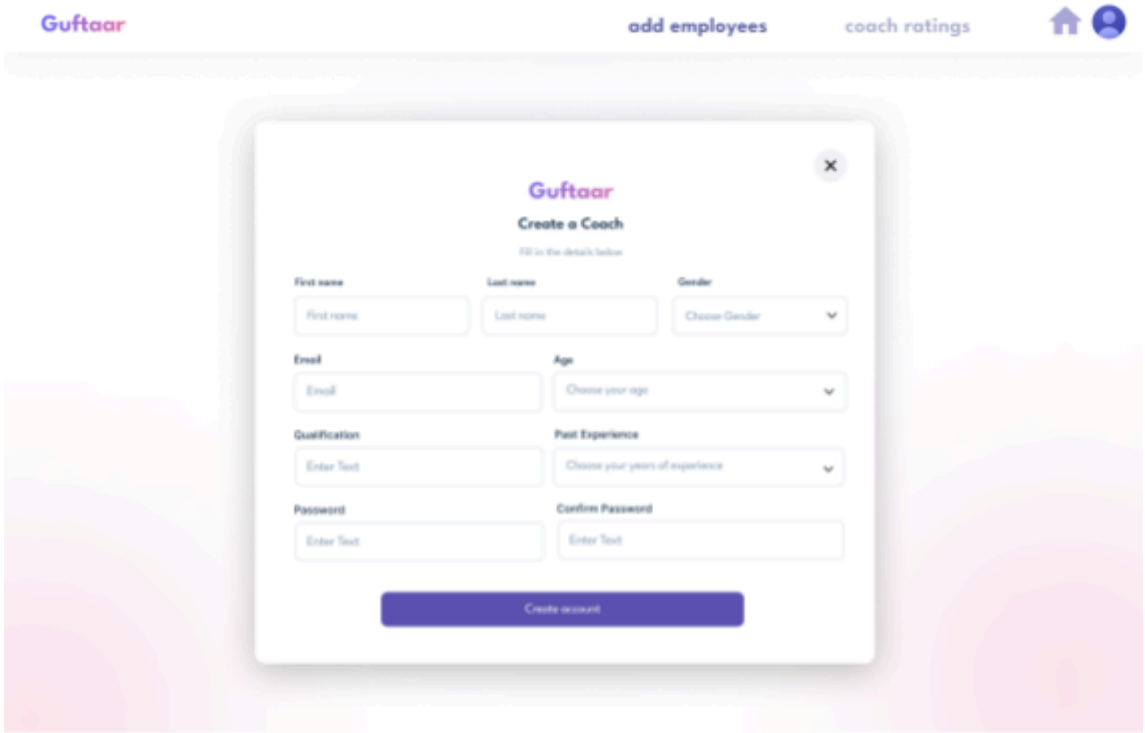


Figure 26 (b): Add Employees

Goal: Figure 26 depicts the Add Employee screen flow. The first screen can be accessed through the dashboard by clicking the Add Employee button. The second screen can be accessed by clicking the Coach button. The goal of these screens is for the admin to be able to systematically add verified coaches to the Guftaar platform.

User Interface: The UI for the Figure 26 consists of the following elements:

Button	Functionality
Admin	Redirect to add admin form page.
Coach	Redirect to add coach form page.
Add Coach Form	HTML form to accept coach details through text and drop-down fields.
Create Account	Add coach to the Guftaar system and append details to the database.
Exit	Redirect to dashboard.

Table 19: UI Components for Figure 26

Although the admin for Guftaar platform is expected to be well-versed with technologies and managing web pages, the UI for Figure 26 stays consistent with the UI design considerations throughout the Guftaar web-app. Usability is improved through highlights and hover-effects, and user-experience is enhanced by using graphics and strategic central placement of buttons. Similarly, accessibility is an important consideration tackled through a readable font and a color-blind neutral color palette. Moreover, ease-of-use is ensured by utilizing drop down menus, where possible, to assist admin with their selection. Guftaar also employs hidden/encrypted passwords to enhance usable security.

Use Case: The screen flow depicted in Figure 26 represents the most important admin use case. Since Guftaar only provides sign-up options to clients, the admin holds the authority to add verified coaches and admins to the Guftaar system. In that regard, the admin provides the required details to create an account for the coach/admin so that they can log into the platform.

9. Coach Dashboard

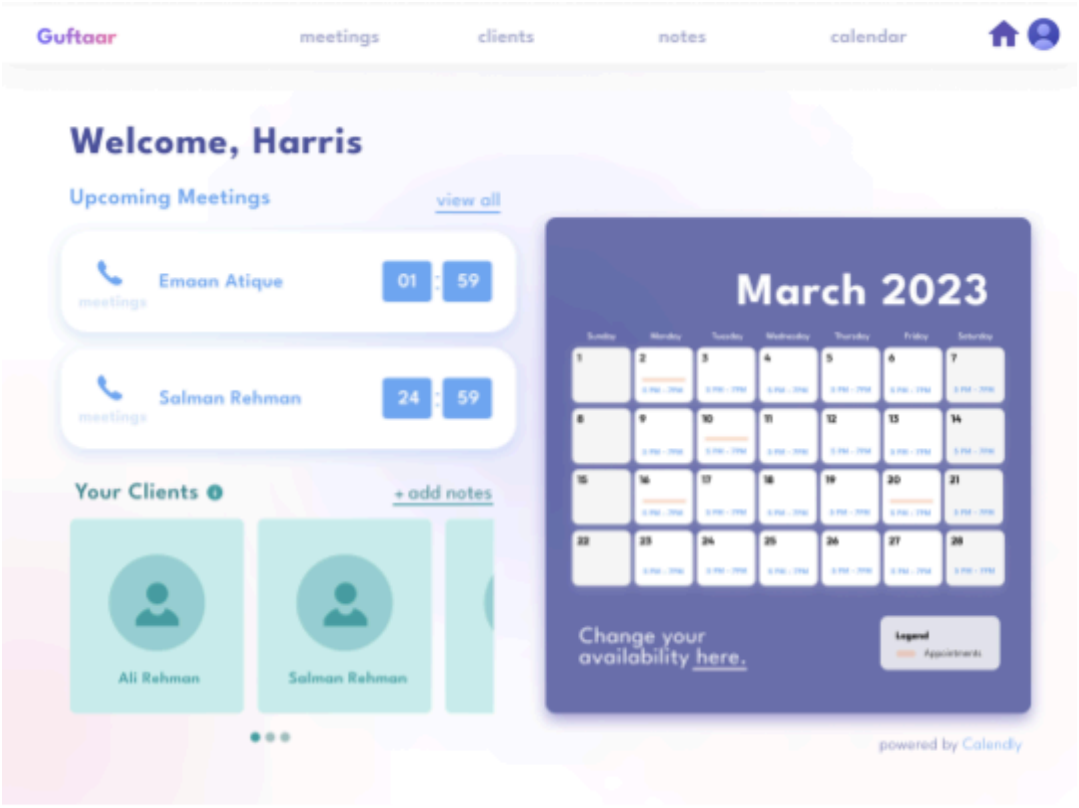


Figure 27: Coach Dashboard

Goal: Figure 27 represents the coach dashboard. The dashboard appears once the coach has successfully logged into the Guftaar platform, and it encapsulates the major functionality available to the coach. The goal of the dashboard is to provide easy accessibility to major functionality. Coaches can view all upcoming meetings, access client profiles for all users who have scheduled a meeting with them, either currently or in the past, can add notes for these clients, and can also view and alter their calendar powered by Calendly. This functionality is available both through dashboard and through navbar, to facilitate the coach.

User Interface: The UI for Figure 27 includes the following elements:

Navbar	
Button	**Functionality**
Meetings	Navigate the coach to a page where they can view all upcoming meetings.
Clients	Navigate the coach to the coaches page where they can see a list of all their clients which they can click on to view client profiles.
Notes	Navigate the coach to the notes page where they can add notes for their clients well as view their previously added notes.
Coach Profile	Display a popup where coach can either logout, or update password

Table 20: UI Components for Navbar

Coach Dashboard	
Button	**Functionality**
Meeting buttons	Navigate the coach to a popup that shows meeting details
View All	Navigate the coach to a page which has a list to all upcoming meetings
Client cards	Open a popup that contains client profile details and coach's previous notes for that client
"Change your availability here"	Navigates the user to the embedded Calendly API where they can alter their available slots for client sessions

Table 21: UI Components for Dashboard

The dashboard UI of Guftaar has been thoughtfully designed with user-friendliness and ease-of-use being the topmost priorities. To ensure a clutter-free and organized interface, the design incorporates compartmentalization and a grid-like layout. The use of drop-shadows and hover-effects in the user interface components further enhances the platform's usability. The calendar gives a macro view of the user's schedule for the next month, while the client cards and the meeting buttons give easy access to essential information, and the complete details can be accessed from the hyperlink buttons. This ensures that the user is not overwhelmed with information, but still can access important information directly from the dashboard.

Apart from usability, Guftaar also prioritizes accessibility for all users. To achieve this, the design incorporates an easy-to-read Sans-serif font family and a color-blind neutral color palette. The use of graphics wherever possible increases the user-friendliness of the platform, allowing users to easily navigate and understand the functions of the dashboard.

Use Case: The dashboard serves as an amalgamation of all coach use cases. It is a portal through which all functionality is made available to the admin with a single click. It is organized in such a way that the use cases are easy to follow.

10. Add Meeting Availability

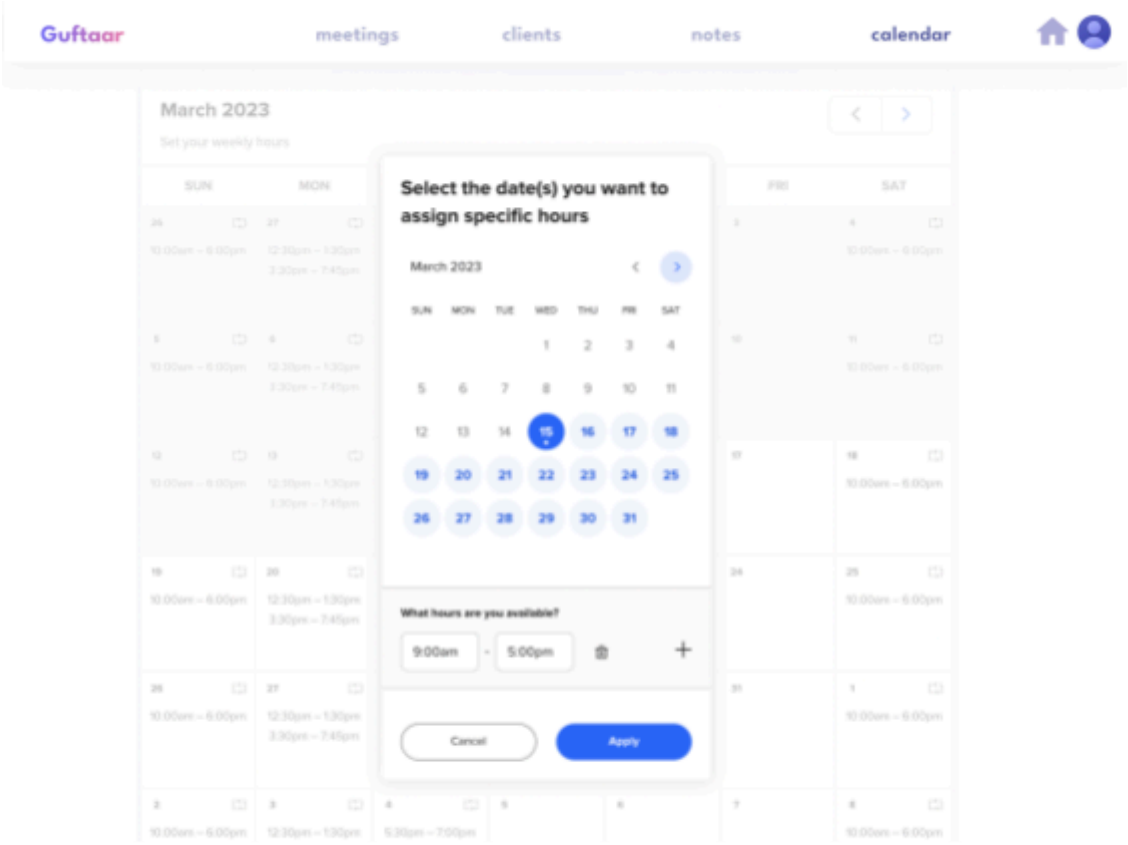


Figure 28: Change Schedule

Goal: Figure 28 shows the integration of the Calendly API to the Guftaar Platform, enabling coaches to add available slots to their schedule for guided therapy sessions with the clients. This page can be accessed directly from the coach dashboard through the 'Change your Availability here' hyperlink.

User Interface: The UI for the Figure 28 consists of the following elements:

Button	Functionality
Change Month Arrow	Select the month for the new slot.
Date	Select the date to add the new slot.
Time	Specify the starting and ending time for the slot.
Discard Slot	Discard the added time slot.
Add New Time Slot	Add another available time slot on the same date.
Apply	Confirm and save changes.
Cancel	Return to the base screen without saving changes

Table 22: UI Components for Figure 28

The UI for Figure 28 primarily comprises the embedded calendly API. Hence, all design and user-experience decisions can be attributed to the embedded Calendly API. The Guftaar platform utilizes this API to manage all types of scheduling. The only component of the Guftaar UI design philosophy is the navbar, which has been covered before.

Use Case: This integration of the Calendly API serves the purpose of adding available meeting slots to the coach schedules so clients can schedule meetings for their guided therapy sessions.

5.4 User interface design rules

Colour Palette

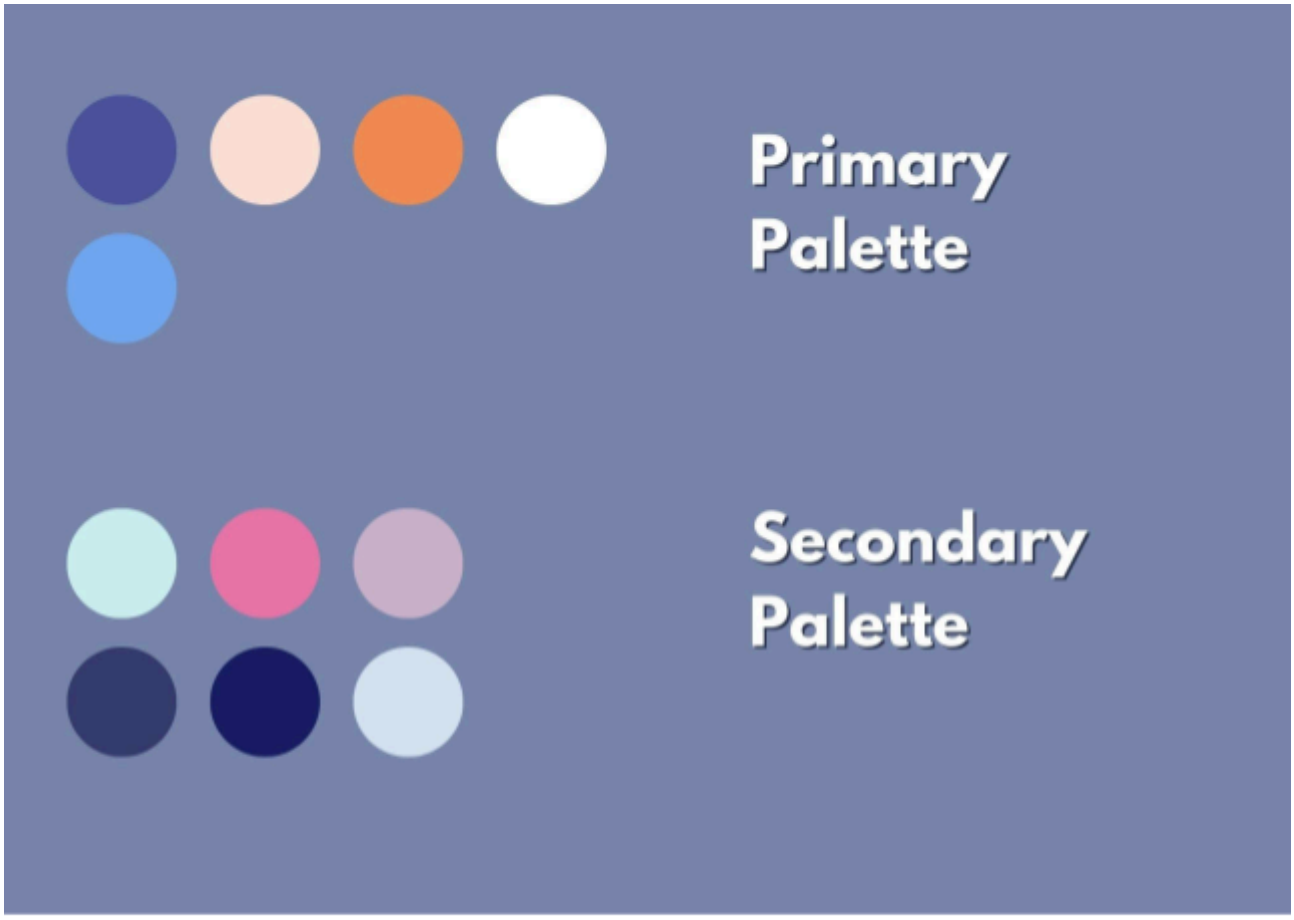


Fig 29: Colour Palette

The choice of a pastel color palette for a speech therapy website has been made with the intention of creating a calming and soothing visual environment that is conducive to effective communication. The utilization of pastel colors featuring varying hues of blues and purples, complemented by additional shades of orange, green, and pink, is recognized for its ability to induce a sense of relaxation in individuals. This characteristic can be particularly advantageous for those seeking speech therapy services. Moreover, the soft and gentle tones of pastel colors align well with the sensitive nature of speech therapy, promoting a sense of comfort and safety for users. This color palette has been carefully selected to complement the goals of the website and enhance the overall user experience.

We incorporated Nielson's and Norman Design principles when designing the frontend of Guftaar. Some main pointers we kept in mind were:

- **Visibility**: We have ensured that all on-screen elements are clearly visible to the user, and that there is no excessive clustering of information that may cause confusion as to how to access and utilize the available features. This has been done with the intention of facilitating a user-friendly experience and optimizing ease of use.
- **Feedback**: We have incorporated a feedback mechanism into our website, specifically in relation to the interactive activities available to users. Our website has been designed to provide users with feedback in a variety of scenarios. For example, upon creating a new account, users will receive feedback regarding whether their account has been successfully created or if there are any errors in the information entered. Similarly, feedback will also be provided to users upon completion of any activities or services offered on the website. This feedback mechanism has been incorporated to ensure that users are aware of their progress and any errors that may need to be corrected, thereby promoting a more positive user experience.
- **Error Handling**:The website implements an error handling mechanism that generates clear error messages in plain language, indicating the specific nature and location of the error so the user can rectify it. This approach is aimed at assisting users in resolving issues and enhancing their overall user experience.
- **Simplicity**: The incorporation of simplicity in our website design can significantly enhance user experience and facilitate ease of use for users. By minimizing complexity and streamlining navigation, users can more effectively interact with the website and perform desired tasks without confusion or frustration. Clear and concise language, intuitive design, and user-friendly features can contribute to a more positive and efficient user experience, ultimately leading to increased user engagement and satisfaction.
- **Typography**: The selection of font (League Spartan), its various font weights, and spacing was made to enhance the readability and clear visual presentation of all website content, aiming to prevent cluttering of information and ensuring a seamless user experience with unobstructed access to all platform features. The different font weights used throughout the website ensure to create visual hierarchy and improve readability. League Spartan's strong, sans-serif writing gives the design a bolder feel, while the typeface's space and alignment make it easier for people to read and understand. The website creates a consistent and professional look by making typography a key component of the design process.
- **Consistency**: To ensure a cohesive and uniform visual experience throughout the website, the color scheme, fonts, content, and themes were maintained consistently across all pages. The intention was to create a seamless appearance that allows for smooth navigation and continuity of the user's interaction with the website.

6 Other Non-functional Requirements

6.1 Performance Requirements

- The web application should respond to the user input within 0.5 seconds. This is necessary to provide a smooth user experience.
- The web application should be able to process up to 50 sign up requests concurrently. This is to ensure that the application can handle traffic spikes without causing significant delays.
- The web application should use encryption to protect user data and isolate every client's activity from their respective assigned coaches unless they provide their consent to share their data.
- The web application should be able to identify at least 10 different speech impediments. This will allow the application to provide users with more accurate results.
- The web application should be able to recognize speech in real-time with an accuracy of at least 95%. This is to ensure that the application can provide accurate results in a timely manner.

6.2 Safety and Security Requirements

- The web application must have a system in place to detect, track, and report any suspicious or malicious activities that occur while the web application is being used.
- The web application must be designed in such a way that users can easily recognize and report any potential risks or dangers of using the web application, including the ability to report any suspicious activity.
- All users must be required to provide complete and accurate personal information (name, age, gender, contact information, etc.) before using the web application via HTML forms and all of our forms on the web app are resistant to SQL injection attacks.
- No third-party source would be able to track what programs a particular user is enrolled in or who the assigned coach is since the security of our application motivates the use of session hosting instead of cookies so there is no way of tracking the activity of a given user. Also, everything is stored in the secure clusters of MongoDB Atlas on their respective servers.
- Some metadata will be collected since our application uses third-party client APIs such as Wordpress, Calendly, and Google Drive – albeit majority of which don't incorporate any advertisements except for Wordpress.
- Passwords are going to be encrypted at the time of signing up, and via parameterized SQL or prepared statements, hashed passwords would get stored to the database keeping the service sanitized from malicious intent.

- Understanding the sensitivity of speech therapy, the app should have role-based access control to ensure that users can only access the features and data that they are authorized to. It will be ensured that a client's progress reports can only be viewed by those granted access to be the client.
- Permission to use the device microphone will be taken from the clients for activities in the *Quick Practice* section.

6.3 Software Quality Attributes

This web application must meet certain quality characteristics in order to serve its purpose for users with speech impediment. The following subsections provide requirements related to the different software quality attributes.

Reliability

The web application must be highly reliable and able to perform consistently and accurately with minimal errors. It should have a high availability rate and be able to withstand system failures. We will include unit testing, integration testing, and user acceptance testing to make sure that the application meets all the requirements and works as expected. Additionally, we will monitor the application for any potential issues that may arise in production. Monitoring can be done using tools such as application performance management (APM) and log analysis. Finally, we as developers should be prepared to quickly respond to any issues that arise and have a plan for addressing and resolving them. By taking these steps, web applications can be made more reliable and less prone to unexpected issues.

Portability

The web application must be portable and able to run on any operating system with minimal modifications. It should be designed in a way that requires minimal effort to deploy it in different environments. By making a web application more portable, developers can create a product that can meet a wide range of customer needs, and make the application more accessible to a greater number of users. Here are a few tips on how to make a web application more portable:

1. Cross-platform technologies, such as HTML5 and JavaScript, can help to make web applications more portable. By utilizing these technologies, we can create applications that are compatible with multiple platforms, such as Windows, Mac OS X, iOS, and Android.
2. Heroku is a cloud-based deploying service which means that by integrating cloud computing, we will be able to make our web applications more portable by making them accessible from any device regardless of the platform. By hosting our service on the cloud, we can ensure that our users can access the application from any device at any time.

3. By using responsive web design and relevant frameworks that enable us to employ this strategy such as TailwindCSS and Bootstrap, we can ensure that their web application looks great and is fully functional at any time.

Adaptability

The web application must be able to adjust and adapt to different user environments and changing requirements. It should be able to adjust to new user requirements and preferences with minimal effort. To make our web application more adaptable, we need to ensure that our service is able to respond quickly and effectively to changes in the environment. First, we will make sure that our code is as modular and extensible as possible. This will give us flexibility to quickly and easily add new features, or make changes to existing ones, without having to rewrite larger portions of code. Second, we will be working on designing and implementing a unit-testing framework that will ensure that any changes to the code do not cause unexpected problems, which will help us identify and address any issues that may arise in a timely manner.

Availability

The web application must be available 24/7 with minimal downtime. It should be designed to be resilient to outages and provide uninterrupted service. First, we will ensure that our web application is optimized for speed and performance which includes a fast web server, effective caching and ensuring the codebase is optimized for performance. Since we will be using pocketbase or firebase to host our server-side, it will take care of all these factors. Second, we will be designing the web app with scalability in mind, i.e., it should be able to handle increased usage and traffic without significant performance degradation. This involves monitoring usage metrics and trends and scaling appropriately. All of these tasks could be performed by the deploying service that we shall be using to host our application.

Flexibility

The web application must be flexible and able to be easily changed or modified. It should be easily extendable and customizable to meet user needs. First, we will consider improving the user experience by adding customization options. This could include giving users the ability to change the font size, color scheme, or layout of the application according to their preferences. Second, we could consider adding more options for integration with other web services. This could include giving users the ability to easily connect their account with other services and allowing them to sync data between them.

Interoperability

The web application must be interoperable with other applications and systems. It should be designed to be compatible with existing systems to ensure seamless integration. To make our web

application more interoperable, we will make sure that we use open standards and industry-recognized technologies that have been trial and tested extensively ensuring our application is compatible with as many systems as possible. Second, we will focus on web APIs that will allow for automated integration with third-party systems such as WordPress, YouTube, Google Drive, etc. Lastly, we will test our application on multiple operating systems and browsers ranging from flagship smartphones to PCs and Laptops. By taking these steps, we can ensure that our web application is interoperable with a wide range of systems and technologies.

Maintainability

The web application must be easy to maintain and update with minimal effort. It should be designed with maintainability and scalability in mind to reduce the effort required for maintenance and upgrade. There are a couple of ways through which we could make our code more maintainable. First, we will create an intuitive design in the form of functional prototypes on Figma which will make the web app easy to understand and navigate. Second, we will use best practices and write code that is easy to read and understand. Third, we will create comprehensive documentation that explains our code and how it works. Fourth, we will set up automated tests to ensure our code is working correctly. Finally, we will use a reliable deployment process that allows us to quickly and easily deploy new versions of our application.

Robustness

The web application must be robust and able to handle various conditions and errors. It should be designed to be resilient to errors and be able to handle unexpected inputs and conditions.We can make our web application more robust by ensuring that our system is fault-tolerant and can handle unexpected changes in the user input and/ or system requirements. Second, we will take parallelism into account in a way that ensures that our application can handle large increases in user demand without requiring major changes into the infrastructure.

Testability

The web application must be easily testable and have automated tests. It should be designed in a way that allows for automated tests to be created and run with minimal effort. Making our web application more testable is important or ensuring that our product is reliable and of high quality. We will be taking several steps to guarantee the testability element of our app. We will start by setting up automated test cases that run on a regular basis. Second, we will use tools such as unit tests and integration tests to test our application. Lastly, we will employ debugging tools to identify and fix any issues we may encounter.

Reusability

The web application must be reusable and able to be used in various contexts. It should be designed to be easily reusable in different scenarios and be modifiable to meet different requirements. First, we will ensure that the code is well-structured and easy to read. This will make it easier to reuse components in the future. Second, we will use a modular approach when designing the application. This will make it easier to replace and update components without having to recreate the entire application. Finally, we will create a library of components that can be used in multiple web sub-applications within our main system.

Usability

The web application must be user-friendly and intuitive, with a focus on ease of use rather than ease of learning. It should be designed to be intuitive and easy to use without requiring extensive knowledge or training. To leverage the usability factor, we will make sure all navigation links are clearly visible and easy to understand. We will add interactive elements to make the user experience more engaging while ensuring that the web page loading time is fast. We will keep the test simple and concise and instead of using text to represent clickable objects, icons would be preferred.

Correctness

The web application must be correct and produce correct results. It should be designed to produce accurate and reliable results in the expected output format. First, we will check our source code for any possible errors and ensure that all the logic is correct and reflects the goals we aim to achieve using our service. We will use static analysis tools to identify any potential errors or issues with the code. Second, we will ensure that our application is properly designed to handle any potential traffic or load.

Appendix A - Group Log

*<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist **ME and the Teaching Assistants** to determine the effort put forth to produce this document>*

1. **Meeting with Assigned TA, February 25 2023**
   Time: 30 Minutes
   Meeting Details:
   Discussed the upcoming Design Deliverable and talked about setting up the Trello Board and gave access for it to the TA. Talked about setting a new time for the weekly meetings with the TA.

2. **Meeting with Assigned TA, March 3 2023**
   Time: 30 Minutes
   Meeting Details:
   Got a general idea about the SDS and the Trello Board so we could divide the work accordingly and start working on it.

3. **Group Meeting, March 12 2023**
   Time: 3 hours
   Meeting Details:
   Researched on what architecture and database fits our website the best. Got done with sections 2 and 3. Discussed the diagrams that had to be made for different sections of the document.

4. **Meeting with Instructor, March 13 2023**
   Time: 30 Minutes
   Meeting Details:
   We were a bit confused as to what architecture suits our implementation model the best and the database that we'll end up using, so we discussed that with Dr. Maryam. We got some insights with regards to the implementation details and what programming language we'll be using.

5. **Group Meeting, March 13 2023**
   Time: 4 hours
   Meeting Details:
   Brainstormed over the basic layout of the diagrams in section 4.1. We got done with much of the component and activity diagrams, and finalized the section 5.

6. **Group Meeting, March 14 2023**
   Time: 3 hours
   Meeting Details:
   Finalized section 4, and completed all the diagrams for the SDS. Discussed the design of all the screens in great detail by making paper prototypes.

7. **Group Meeting, March 15 2023**
   Time: 4 hours
   Meeting Details:
   Discussed the horizontal and vertical flows of all the screens extensively and the designs of and finalized the majority of the screens.

8. **Group Meeting, March 16 2023**
   Time: 5 hours
   Meeting Details:
   Went over the whole SDS document thoroughly and finalized all screens and diagrams. added any more little details that had to be added and discussed and finalized everything with the group to make the document ready for submission.

Appendix B – Contribution Statement

Name	Contributions in this phase	Approx. Number of hours
Emaan Atique	2.1, 2.3, 4.1, 4.2, 4.4, 4.5	40
Saad Sher Alam	1.2, 4.1, 5.1, 5.2, 5.3 (worked on the screen descriptions), 4.1, 4.2	40
Bakhtawar Ahtisham	3.1, 3.2, 3.3, Appendix A, 2.4, 5.4 (brainstormed over it), 6.1	40
Harris Ahmad	3.1, 3.2, 3.3, 4.3, 4.4, 4.1, 6.2, 6.3	40
Emaan Bilal	1.1, 2.2, 4.1, 4.2, 4.4, 5.3	40
Rumessa Shah Jahan	2.1, 2.2, 2.3, 2.4, 2.5, 4.1, 5.4, Appendix A	40