

**A  
PROJECT REPORT  
ON**

**“KeyDash”**

**Submitted for the partial fulfillment of the Degree of Bachelor of Engineering in  
Computer awarded by Pokhara University**



**SUBMITTED BY:**

Anurag Ale (22070446)

Britant Shrestha (22070449)

Manish Shrestha (22070452)

Ranjana Dhakal (22070457)

**SUPERVISOR:**

Er. Anil Timilsina

**DEPARTMENT OF COMPUTER ENGINEERING  
POKHARA ENGINEERING COLLEGE**

Phirke, Pokhara-8

2025

## **CERTIFICATE**

**Date: 2025/07/04**

This is to certify that the project entitled “**KeyDash**” has been carried out by Anurag Ale (22070446), Britant Shrestha (22070449), Manish Shrestha (22070452) and Ranjana Dhakal (22070457) in partial fulfillment of the degree of Bachelor of Computer Engineering of Pokhara University, during the academic year 2025. To the best of our knowledge and belief, this work has not been submitted elsewhere for the award of any other degree and thus has been accepted.

-----  
**Er.**  
**External Examiner**

-----  
**Er. Anil Timilsina**  
**Supervisor**

-----  
**Er. Suraj Basant Tulachan**  
**HOD, Computer & Electronics**

-----  
**Er. Krishna Ghimire**  
**Co-Ordinator, Research Management Cell**

## ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude to everyone who supported and guided us throughout the completion of our project. First and foremost, we are profoundly thankful to our respected supervisor, **Er. Anil Timilsina**, for his expert guidance, continuous encouragement, and constructive suggestions at every stage of this project. His patience and dedication to our academic growth significantly shaped both the technical and practical direction of our work.

We also express our sincere appreciation to the **Department of Computer Engineering** for providing a strong academic foundation and the essential resources that made this project achievable. The department's curriculum, lab facilities, and regular project evaluations were instrumental in our learning process.

We would like to acknowledge the contributions of all faculty members and technical staff, whose teachings and timely assistance helped us overcome many challenges during the development phase. Their expertise and support added great value to our learning experience.

Special thanks to our classmates and friends who helped with early testing, shared ideas, and gave us constructive feedback that improved the overall quality and usability of our

Lastly, we want to recognize the collective effort of our group. This project would not have been possible without our collaboration, mutual understanding, and commitment toward achieving a common goal.

Anurag Ale (22070446)

Britant Shrestha (22070449)

Manish Shrestha (22070452)

Ranjana Dhakal (22070457)

## Table of Contents

<b>Certificate.....</b>	<b>i</b>
<b>Acknowledgement.....</b>	<b>ii</b>
<b>Abstract .....</b>	<b>1</b>
<b>Chapter 1 Introduction .....</b>	<b>2</b>
1.1 Background .....	2
1.2 Theory .....	2
1.2.1 React Framework .....	2
1.2.2 Supabase Backend .....	2
1.2.3 PostgreSQL .....	3
1.2.4 User Authentication .....	3
1.2.5 Cloud Storage.....	3
1.2.6 Data Modeling .....	3
1.2.7 State Management in React .....	3
1.2.8 Version Control with GitHub .....	4
1.2.9 Security Considerations .....	4
1.3 Project Category (Web Application) .....	4
1.4 Detailed Problem Definition .....	5
1.5 Presently Available System for the Same Purpose.....	5
1.6 Objective of the Project.....	6
1.7 Scope.....	6
<b>Chapter 2 Literature Review.....</b>	<b>7</b>
2.1 Introduction.....	7
2.2 Content Review .....	7
2.3 Traditional Review .....	7

2.4 Our Proposed System.....	7
<b>Chapter 3 Analysis.....</b>	<b>8</b>
3.1 Project Management .....	8
3.1.1 Initial Investigation .....	8
3.1.2 Feasibility Analysis.....	8
3.1.3 Cost Benefit Analysis.....	9
3.2 Requirement Analysis .....	9
<b>Chapter 4 Design .....</b>	<b>11</b>
4.1 Software Requirements Specification.....	11
4.1.1 Functional Requirements .....	11
4.1.2 System Specification.....	11
4.1.3 Non-functional Requirements.....	12
4.2 Risk Assessment.....	12
<b>Chapter 5 System Modeling .....</b>	<b>14</b>
5.1 Block Diagram .....	14
5.2 UML Diagram .....	15
5.2.1 UML Use Case Diagram.....	15
5.3 Flowcharts .....	16
5.3.1 System Flowchart.....	16
5.4 System Workflow .....	17
5.4.1 Registering users.....	18
5.4.2 Log in option.....	18
5.4.3 View mode and respective mode .....	18
5.4.4 View Leaderboard.....	18
5.4.5 View Scoreboard.....	18
5.4.6 Profile Section.....	19
5.5 Data Flow Diagram.....	20

5.5.1	Context Level DFD.....	20
5.5.2	Level-1 DFD .....	20
5.6	Database Design.....	21
5.6.1	ER (Entity Relationship) Diagram.....	21
5.6.3	Supabase Database Table for users .....	22
5.6.4	Supabase As a Database .....	22
<b>Chapter 6 Coding .....</b>		<b>25</b>
6.1	Hardware Specification.....	25
6.2	Platform.....	25
6.3	Technology Used.....	25
6.3.1	React Technology.....	25
6.4	Programming Language Used.....	26
6.4.1	JavaScript Programming Language .....	26
6.4.2	SQL (Structured Query Language) .....	27
6.5	Software Tools Used .....	27
6.5.1	Visual Studio Code .....	27
6.5.2	Git and GitHub.....	27
6.5.3	Web browser .....	27
6.5.4	Adobe Illustrator .....	28
6.5.5	Vercel.....	28
6.6	Coding Style Followed.....	28
<b>Chapter 7 Testing.....</b>		<b>30</b>
7.1	Test Plan .....	30
7.2	Test Cases.....	30
7.2.1	Testing Cases Summary .....	30
7.2.2	Sample Test Cases.....	31
<b>Chapter 8 Limitations And Recommendations .....</b>		<b>37</b>

8.1 Limitations .....	37
8.2 Recommendation .....	37
<b>Chapter 9 Conclusion.....</b>	<b>38</b>
9.1 Conclusion .....	38
<b>REFERENCES .....</b>	<b>39</b>

## FIGURE INDEX

S.N.	FIGURE	Page No.
1.	Figure 5.1: Block Diagram of the System .....	14
2.	Figure 5.2: UML Use Case Diagram .....	15
3.	Figure 5.3: System Flowchart.....	16
4.	Figure 5.4: System Workflow .....	17
5.	Figure 5.5: Context Level DFD .....	20
6.	Figure 5.6: Level-1 DFD.....	20
7.	Figure 5.7: ER Diagram.....	21
8.	Figure 7.1: Test case 1.....	31
9.	Figure 7.2: Test case 2.....	32
10.	Figure 7.3: Test case 3.....	33
11.	Figure 7.4: Test case 4.....	34
12.	Figure 7.5: Test case 5.....	35
13.	Figure 7.6: Test case 6.....	36

## TABLE INDEX

S.N.	TABLE	Page no.
1.	Table 5.1: Supabase Table Profile.....	22
2.	Table 5.2: Supabase Table Leaderboard_sentence.....	23
3.	Table 5.3: Supabase Table Leaderboard_timed.....	23
4.	Table 7.1: Testing Cases Summary .....	29
5.	Table 7.2: Test Case 1 .....	31
6.	Table 7.3: Test Case 2 .....	31
7.	Table 7.4: Test Case 3 .....	31
8.	Table 7.5: Test Case 4.....	32
9.	Table 7.6: Test Case 5.....	33
10.	Table 7.7: Test Case 6.....	34



## APPENDICES

Appendix A: Login screen .....	39
Appendix B: Sentence mode.....	39
Appendix C: Timed mode .....	40
Appendix D: Leaderboard.....	40
Appendix E: Profile Section .....	41
Appendix F: Profile Update Setting.....	42
Appendix G: Backend Database .....	43

## ABSTRACT

“**KeyDash**” is an interactive and user-friendly web application designed to measure typing speed and accuracy in real-time. It is developed as a project for the partial fulfillment of degree of Bachelor of Engineering in Computer Engineering of Pokhara University. The project has been developed by us with a great support of our college administration, faculties and friends and is liable to be submitted to the Department of Computer Engineering College, Phirke – 8, Pokhara, Nepal.

“**KeyDash**” provides an engaging platform for users to test and improve their typing skills. The application features a dynamic interface with customizable test durations, real-time performance metrics (Words Per Minute - WPM, accuracy, and error count), and a results history to track progress over time. It is built using modern web technologies and it ensures a seamless and responsive experiences across devices. The project emphasizes clean code practices, efficient algorithms for performance calculation, and an intuitive UI/UX design. It serves as both an educational tool, demonstrating proficiency in full-stack web development.

The software development team is really appreciative to our teacher, supervisor, and friends for their invaluable assistance and cooperation in bringing this project to life. The process of creating this project has been fantastic, and we hope that our users will feel the same way.

# Introduction

## Background

In the current digital era, typing accuracy and speed are critical abilities for professionals, students, and regular computer users. Typing efficiently improves workflow overall, increases productivity and cuts down on time spent on documentation. By giving users immediate feedback in performance indicators including words per minute (WPM), accuracy and error rates, typing speed testers frequently used to evaluate and enhance these abilities.

Current typing test programs frequently don't include progress-tracking capabilities, real-time analytics, or a contemporary, user-friendly design. Many are too complicated or too simple, which deters users from using them. A well-designed, feature-rich, and responsive typing speed tester is required in order to provide a smooth user experience and assist users in monitoring their progress over time.

## Theory

### React Framework

React is an open-source JavaScript library developed by Meta (formerly Facebook) for building dynamic and interactive user interfaces. It follows a component-based architecture, enabling developers to create reusable UI elements that update efficiently via a virtual DOM. For KeyDash, React was chosen due to its performance optimizations, extensive ecosystem (e.g., hooks), and compatibility with modern web development practices. Its declarative nature simplifies rendering real-time typing metrics (e.g., WPM, accuracy) during tests.

### Supabase Backend

Supabase is an open-source Backend-as-a-Service (BaaS) platform that provides PostgreSQL database management, authentication, and storage APIs. KeyDash leverages Supabase to handle user data, test results, and authentication, eliminating the need for custom server-side code. Supabase's RESTful and real-time APIs ensure

seamless integration with the React frontend, enabling features like historical performance tracking and multi-device synchronization.

## PostgreSQL

PostgreSQL is a robust, open-source relational database system powering Supabase. It stores structured data for KeyDash, including user profiles, test histories, and typing analytics. PostgreSQL's scalability and ACID compliance ensure data integrity for high-frequency operations (e.g., saving test results) while supporting complex queries for generating user progress reports.

## User Authentication

KeyDash implements secure user authentication via Supabase Auth, offering email/password sign-up and OTP verification. This ensures only authenticated users can save and compare their typing test results across sessions. Role-based access control (RBAC) restricts sensitive operations, such as modifying test parameters or deleting accounts.

## Cloud Storage

Supabase storage manages images given by users for profile setting. Images are stored securely within supabase storage.

## Data Modeling

Data modeling involves designing the structure of a database to ensure efficient storage and retrieval of information. In KeyDash, a relational data model was created to map best wpm, id, scores, display name, accuracy, etc.

## State Management in React

KeyDash uses React's hooks to manage global state, including:

- Real-time test metrics (keystrokes, errors).
- User authentication status.

## Version Control with GitHub

KeyDash uses Git for local version control and GitHub as the remote repository to enable collaborative development. KeyDash uses Git for local version control and GitHub as the remote repository to enable collaborative development. The project leverages CI/CD (Continuous Integration and Continuous Deployment) pipelines to automate testing and deployment workflows with Vercel, ensuring code reliability and efficient updates.

## Security Considerations

Security web application is critical, especially when handling user-generated content. KeyDash employs several security practices including authentication checks before file uploads, secure storage of profile images and table policy. By leveraging Supabase's built-in security features, the application maintains a high level of trust and data integrity among its users.

## Project Category (web application)

KeyDash is developed as a web application to provide universal accessibility across devices, including desktops, laptops, and tablets, without requiring platform-specific installations. Unlike mobile apps, a web-based approach ensures instant access via browsers, making it ideal for users who frequently switch between devices (e.g., home computers, library workstations, or shared devices).

The goal of KeyDash is to offer a seamless typing practice experience with real-time analytics, leaderboards, and progress tracking. By choosing a responsive web design, we prioritize:

- **Cross-Platform Compatibility:** Works on any device with a modern browser (Chrome, Firefox, Edge).
- **Ease of Updates:** No app store approvals needed; users always access the latest version.
- **Performance:** Leverages React's virtual DOM for smooth rendering of typing tests and animations.

Built with React.js and Supabase, KeyDash delivers:

- Zero Setup: Users can start typing tests immediately—no downloads or installations.
- Cloud Sync: Progress and scores are saved securely and sync across devices via Supabase backend.
- Keyboard-First UX: Optimized for physical keyboards (common on desktops/laptops), enhancing typing accuracy measurements.

Choosing a web application wasn't just a technical decision—it aligns with how typists and professionals actually practice. Many users prefer full-sized keyboards for training, and web apps eliminate friction for quick practice sessions during work breaks or study time. By making KeyDash browser-based, we meet users where they already are: online, logged in, and ready to improve their typing skills.

## Detailed Problem Definition

The following are the main obstacles to creating a useful typing speed tester:

- Real-time, accurate evaluation of typing speed (WPM) and error rate.
- Designing a user interface that is both visually appealing and intuitive.
- Implementing a system to store and retrieve user performance history.
- Ensuring responsiveness and cross-device compatibility.

KeyDash aims to solve these challenges by utilizing modern web technologies to create a reliable, engaging, and scalable typing assessment tool.

## Presently Available System for the Same Purpose

Currently, users looking to practice or evaluate their typing skills must rely on various disconnected solutions, each with significant limitations. While websites like 10FastFingers offer basic typing tests, they lack comprehensive features such as personalized progress tracking, customizable test options, or user accounts to save results across devices. Desktop applications like TypingMaster provide structured lessons but require installations and often miss real-time feedback and competitive elements. Browser-based tools and mobile apps focus on specific aspects like

algorithmic text generation but fail to integrate social features or centralized performance tracking. Informal methods, such as practicing in document editors, offer no measurable metrics, while gaming platforms prioritize entertainment over skill development. These fragmented solutions share common drawbacks: they don't combine speed testing with accuracy analytics and social competition, provide poor organization of results, and offer minimal customization options. KeyDash addresses these gaps by delivering a unified web platform with secure accounts, real-time analytics, and shareable leaderboards, creating a more complete typing practice solution.

## Objective of the Project

- KeyDash offers real time wpm and diverse text sources, allowing users to tailor difficulty levels to their typing skills.
- KeyDash provides leaderboard to compete between users.

## Scope

The scope of KeyDash depends on the successful implementation of an intuitive UI, engaging UX, and features that cater to both casual typists and professionals. Like many popular tools, its effectiveness lies not in complex technology but in delivering a seamless user experience that keeps people coming back to improve their typing skills.

The scope of the KeyDash application is:

- KeyDash provides real-time typing tests in timed, sentence-based, and custom text modes to assess accuracy, speed (WPM), and errors. With comprehensive analytics and performance history, it also monitors user progress.
- KeyDash includes competitive leaderboards where users can compare their scores globally or with friends, fostering motivation and community engagement.

# Literature Review

## Introduction

Typing speed and accuracy are critical skills in the digital era, impacting productivity and efficiency in academic and professional settings. Research indicates that real-time feedback and structured practice significantly enhance typing proficiency by reducing cognitive load and reinforcing muscle memory [1], [2]. Modern typing tutors leverage interactive methods to improve engagement and learning outcomes [5].

## Content Review

Existing typing test platforms, such as 10FastFingers and TypingClub, offer standardized tests with performance metrics like Words Per Minute (WPM) and accuracy [3], [4]. However, many lack customization options, localized language support, or detailed progress tracking, limiting their effectiveness for structured learning [6].

## Traditional Review

Students and professionals often rely on basic typing tests or manual practice methods, which lack real-time analytics and personalized feedback [6]. While some institutions integrate typing exercises into coursework, these are rarely systematic or data-driven [6]. Additionally, many existing solutions do not support multi-language input or fail to store historical performance data, making long-term progress tracking difficult [6].

## Our Proposed System

KeyDash addresses these gaps by offering a modern, adaptive typing speed tester with real-time WPM and accuracy tracking. Unlike generic platforms, KeyDash supports customizable test durations, multiple input languages, and supabase-backed user profiles for progress monitoring. Built with HTML, CSS, and JavaScript, it ensures accessibility across devices while providing a responsive, user-friendly interface. By focusing on structured practice and performance analytics, KeyDash bridges the divide between casual typing tests and skill-based mastery [7].



# Analysis

## Project Management

The successful creation of KeyDash, a typing speed tester intended to assist users in increasing their typing efficiency, depended heavily on efficient project management. The project lifecycle is described in this part, including requirement analysis, cost-benefit analysis, feasibility analysis, and early research.

### Initial Investigation

The idea for KeyDash emerged from recognizing the need for a modern, user-friendly typing speed tester that provides real-time feedback and progress tracking. Existing solutions often lacked customization, multi-language support, or detailed analytics. A preliminary survey among students and professionals confirmed a demand for a tool that combines accuracy measurement (WPM), error analysis, and historical performance tracking. Based on these findings, KeyDash was proposed to offer an intuitive, feature-rich typing assessment platform.

### Feasibility Analysis

To ensure the project's viability, four key feasibility areas were evaluated:

- **Technical Feasibility**

The project is technically feasible, leveraging HTML, CSS, and JavaScript for frontend development, ensuring cross-platform compatibility. Supabase was chosen for backend services (authentication and data storage) due to its scalability, real-time updates, and ease of integration. These technologies are well-documented and widely used, making development efficient.

- **Economic Feasibility**

The project is economically viable. All tools and frameworks used (Visual Studio Code, Supabase free tier, GitHub Pages for hosting) are free or open-source. Development was conducted on existing hardware, eliminating additional costs. Future expenses, if any, would involve optional Supabase plan upgrades for increased user traffic.

- **Schedule Feasibility**

A clear development timeline was set and followed throughout the project. Over the course of several weeks, the complete system was created, including stages such as planning, designing, coding, testing, and final deployment. The project was time-feasible because it was finished inside the academic calendar.

- **Legal Feasibility**

The application does not violate any intellectual property or institutional regulations. It is intended for educational use, and all materials shared on the platform are user-uploaded. The system includes privacy-respecting authentication and does not collect or misuse sensitive personal data.

## Cost Benefit Analysis

The project's costs were minimal:

- Development tools: Free (VS Code, Supabase, GitHub).
- Testing: Conducted on personal devices.
- Deployment: Free via Vercel.
- Documentation: Physical copy (10 per page costs).

## Requirement Analysis

To ensure KeyDash is functional, secure, and user-centric, the following requirements were prioritized:

- **Eligibility**

Only authenticated users (via email/password or Google login) can save progress. This prevents misuse and ensures personalized tracking.

- **Uniqueness**

Each test session produces a unique result that is linked to the user's account. This stops duplicate entries and lets for accurate progress analysis.

- **Privacy**

Supabase securely stores and encrypts user data, including email and WPM

history. There is no implementation of permanent text storage or keystroke logging.

- **Transparency**

Users can view their historical performance but are not tracked beyond aggregated metrics (WPM, accuracy). No intrusive data collection is performed.

# Design

## Software Requirements Specification

KeyDash was developed using industry-standard tools including Visual Studio Code as the primary IDE for frontend coding (HTML/CSS/JavaScript), Figma for UI/UX design prototyping, Supabase for backend authentication and real-time data management, and GitHub Pages for free deployment. These technologies were carefully chosen for their accessibility, seamless integration capabilities, and developer-friendly features to optimize the development workflow and ensure project success.

### Functional Requirements

The functional requirements of KeyDash are:

- User Authentication: The system shall allow users to register and log in via email/password or Google authentication.
- Real-Time Typing Test: The app shall measure typing speed (WPM), accuracy, and error count in real time.
- Test Customization: Users shall select test duration (30s, 60s, 120s) and text source (random words, quotes, or custom input).
- Progress Tracking: Authenticated users shall view their historical test results, including WPM trends and accuracy improvements.
- Responsive Design: The interface shall adapt seamlessly to desktops, tablets, and mobile devices.

### System Specification

#### 1. Software Requirements

- Browser
- Prototyping: Figma
- Logo design: Adobe illustrator
- User Interface: React
- Back End: Supabase

- Database: PostgreSQL

## 2. Hardware Requirements

- Device that supports web service.

## Non-functional Requirements

The following are some non-functional requirements of our system:

- Performance: The app shall load quickly (<2s) and run smoothly, even on low-end devices.
- Usability: The interface shall be intuitive, with minimal navigation steps to start a test or view results.
- Scalability: Supabase backend shall handle increasing users and data without performance degradation.
- Security: User data (email, test history) shall be encrypted and stored securely via Supabase Authentication.
- Maintainability: Modular JavaScript code shall allow easy updates and feature additions.
- Portability: The web app shall function across browsers (Chrome, Firefox, Edge) and devices.
- Reliability: Real-time Supabase synchronization shall ensure data consistency, with graceful error handling.

## Risk Assessment

Developing a typing speed tester like KeyDash involves technical, operational, and user-related risks that must be proactively managed.

Technical Risks:

A major concern is server dependency, as KeyDash relies on Supabase for authentication and data storage. Any Supabase service disruption could temporarily disable user logins or progress tracking. Another risk is inaccurate WPM/accuracy

calculations, which could undermine user trust. Rigorous algorithm testing and real-time validation checks ensure consistent performance metrics.

#### Security & Data Risks:

Unauthorized access to user profiles or test history could compromise privacy. Supabase's built-in authentication and encryption minimize this risk, while strict access controls prevent data leaks. Data loss from accidental deletions or sync failures is mitigated through Supabase's automatic backups and versioning.

#### User Adoption & Behavior:

Low engagement may occur if users find the interface unintuitive or lack motivation to track progress. KeyDash counters this with a gamified design (e.g., progress charts, achievement badges) and seamless onboarding.

#### Accessibility & Compatibility:

Disparities in device performance or browser support could create inconsistent experiences. The app is rigorously tested across browsers (Chrome, Firefox, Edge) and screen sizes, with fallback options for older devices. Offline functionality allows practice sessions without internet, though analytics require reconnection.

By addressing these risks through robust architecture, user-centric design, and fail-safes, KeyDash ensures reliability and long-term usability for typists at all skill levels.

# System Modeling

## Block Diagram

In block diagram for android application, it works as follows

- i. User signup with the email and password.
- ii. User sign in with the registered email and password or google.
- iii. User select gamemode.
- iv. User select difficulty and start typing.
- v. User view leaderboard.
- vi. User view and manage profile.

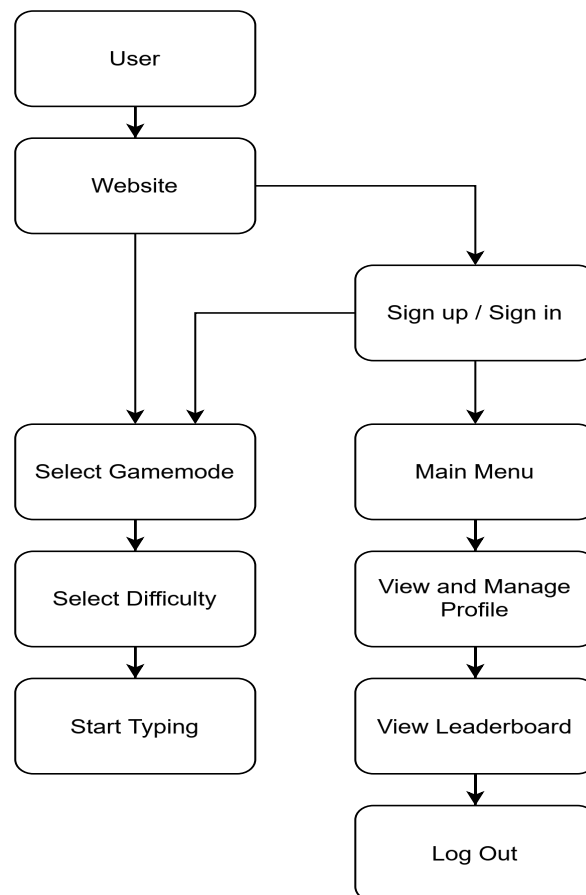


Figure 5.1: Block Diagram of the System

## UML Diagram

### UML Use Case Diagram

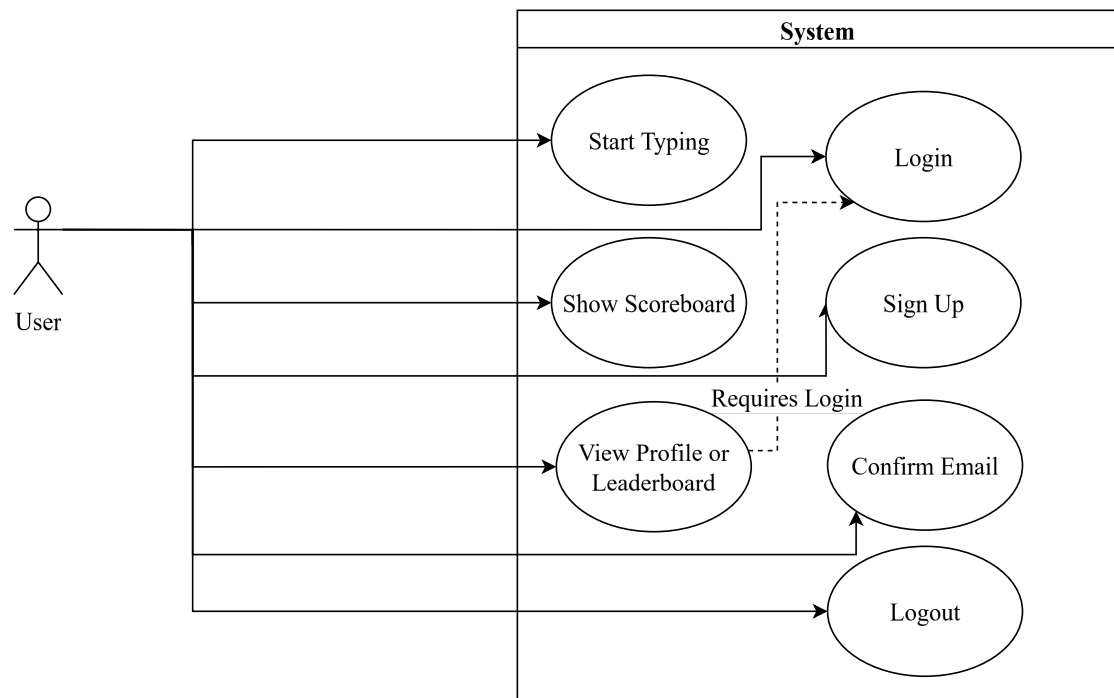


Figure 5.2: UML Use Case Diagram



## Flowcharts

### System Flowchart

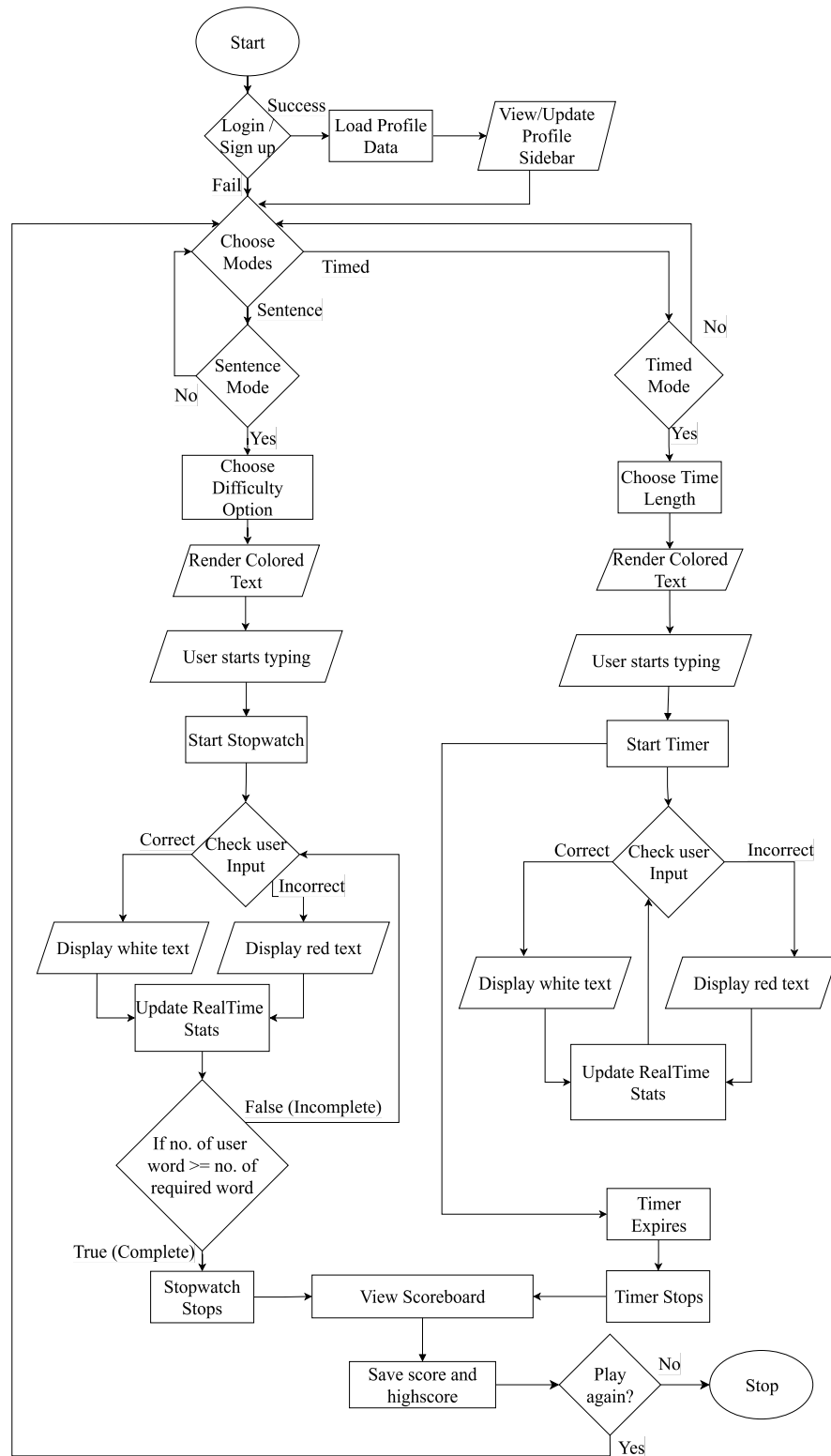


Figure 5.3: System Flowchart

The flowchart of KeyDash illustrates the user interaction flow in the typing speed tester, starting with login/signup authentication, followed by profile data loading and mode selection (Timed or Sentence). In Sentence Mode, the user selects a difficulty level, and the system renders colored text for typing; correct inputs turn text white, while errors turn it red, with real-time stats updating dynamically. The process continues until the user completes the sentence, triggering score.

## System Workflow

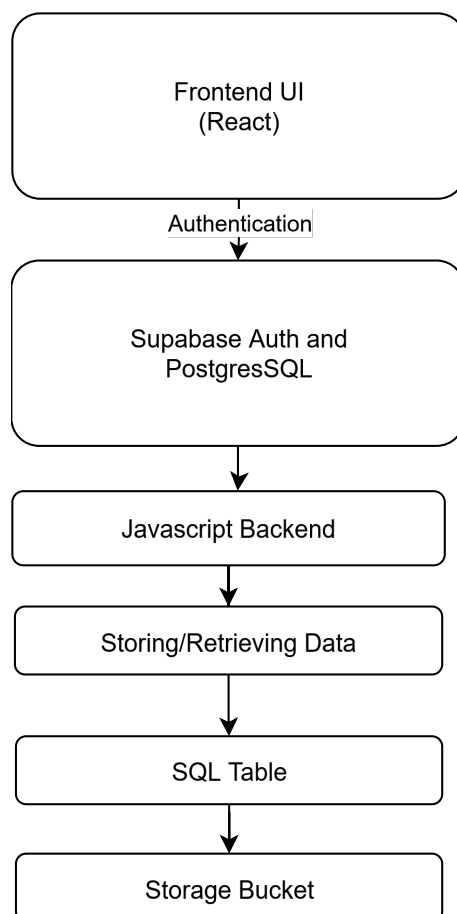


Figure 5.4: System Workflow

## Registering Users

User with a valid email address is capable of registering in the system. Supabase provides its own authentication procedure to make sure whether the email can be registered or not. It sends the confirmation link in mail to complete the registration procedure. After following the confirmation link it directly log in to the system.

## Log In Option

User with the confirmed email is able to login to the system. If the valid email is entered only then the user is able to gain access in the system, if email is not confirmed then user is prompted to confirm the email. Our system provides passwordless login. User are also provided with the option of continuing with their google account i.e. Gmail account.

## View modes and respective levels

User after logging into the system and choose the level they want or refers. User can get access to the difficulty modes even if they haven't logged in the system, it is not mandatory. But without it they cannot record or track their progress.

## View Leaderboard

The authenticated users can look up to the leaderboard of the modes and compete to other users in different modes with their respective levels. Every mode have their own leaderboard where the top score are mentioned on the basis of key metrics (WPM, accuracy, etc.).

## View Scoreboard

After user completes typing in the selected mode, a scoreboard appears. It includes WPM, accuracy, mistakes, characters, etc. User can view it after they completes their task.

## Profile Section

On the profile section, users are able to track back their all of the past session. They can view their best score, average accuracy and total number of sessions with their record.

## Data Flow Diagram

### Context Level DFD

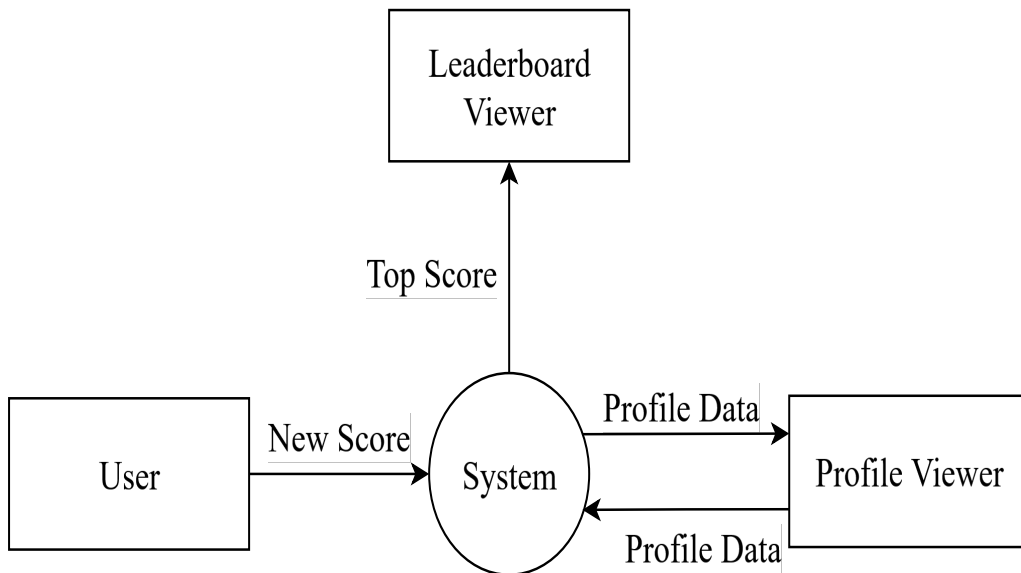


Figure 5.5: Context Level DFD

### Level-1 DFD

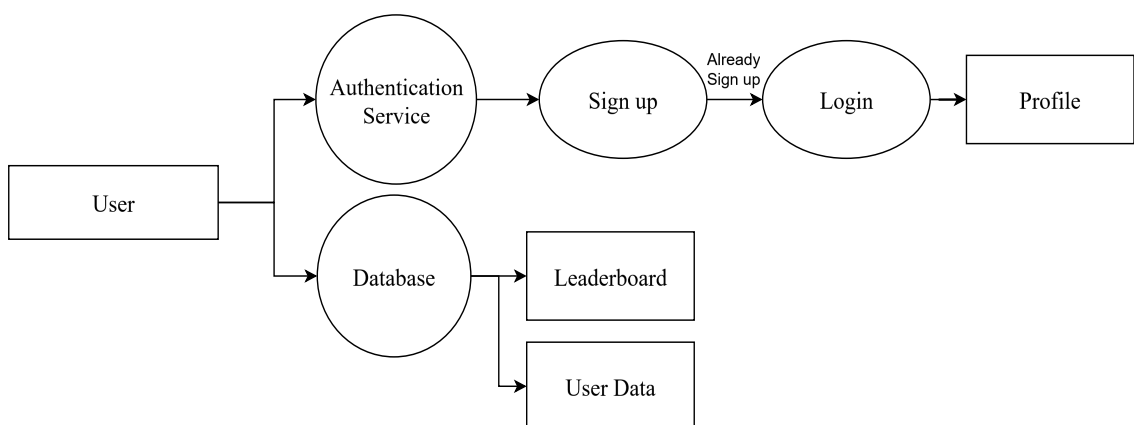


Figure 5.6: Level-1 DFD

## Database Design

### ER (Entity Relationship) Diagram

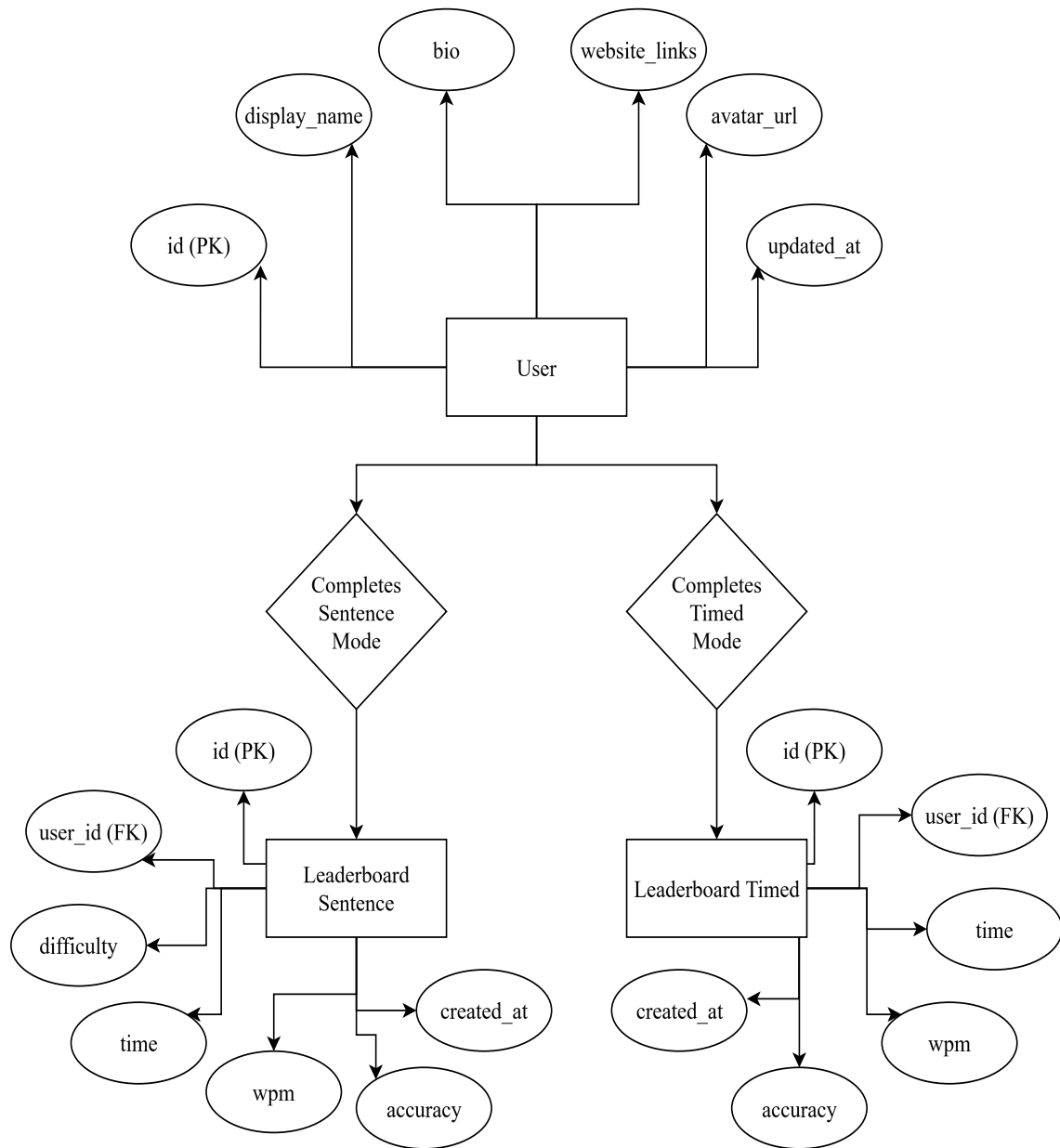


Figure 5.7: ER Diagram

### Supa base Database Table for users

Supa base Database table basically contain 3 table: profile, leaderboard\_sentence and leaderboard\_timed.

S.N.	Field	Data Type	Description
1	id	UUID	Id of the user identifier
2	display_name	Text	Displays the user name
3	bio	Text	Display the user biodata
4	twitter	Text	Twitter info
5	gitHub	Text	GitHub info
6	linkedIn	Text	LinkedIn info
7	instagram	Text	Instagram info
8	created_at	Time Stamp	Timestamp when user data was recorded

Table 5.1: Supa base Table Profile

S.N.	Field	Data Type	Description
1	id	UUID	Id of the user identifier
2	user_id	UUID	Id of the user
3	difficulty	Text	Difficulty level of sentence mode
4	time	Numeric	Time taken to complete the entire sentence
5	wpm	Numeric	Word per meter
6	accuracy	Numeric	Accuracy of typing
7	created_at	Time Stamp	Timestamp when user data was recorded

Table 5.2: Supa base Table Leaderboard\_sentence

S.N.	Field	Data Type	Description
1	id	UUID	Id of the user identifier
2	user_id	UUID	Id of the user
4	time	Numeric	Time option is provided
5	wpm	Numeric	Word per meterin t
6	accuracy	Numeric	Accuracy of typing
7	created_at	Time Stamp	Timestamp when user data was recorded

Table 5.3: Supa base Table Leaderboard\_timed



## Supabase As a Database

Supabase was selected as the backend solution for KeyDash due to its simplicity, flexibility, and suitability for real-time applications. Supabase is an open-source Backend-as-a-Service (BaaS) that provides a complete backend ecosystem built on top of PostgreSQL. It includes features such as authentication, real-time database, file storage, etc making it ideal for a dynamic typing speed tester like KeyDash.

A key advantage of Supabase is its JavaScript/React SDK, which enables straightforward integration with frontend frameworks. This allows for efficient implementation of user authentication, data storage, and real-time updates without complex server management. Built on PostgreSQL, Supabase ensures high performance, scalability, and relational database capabilities, essential for managing user profiles, typing statistics, and leaderboard rankings.

In the context of KeyDash, Supabase is used for:

- Authentication: Secure user registration, login, and session management via email/password.
- Database: Stores user profiles (name, email) and typing performance metrics (WPM, accuracy, test history).
- Real-time Updates: Instantly syncs leaderboard data and personal best scores across all users.

Supabase's user-friendly dashboard simplifies database schema management, query execution, and real-time monitoring—accelerating development and debugging. KeyDash's use of Supabase results in a low-maintenance, scalable, and secure backend that improves user experience and developer productivity.

## Coding

### Hardware Specification

Since KeyDash is a web-based application, it does not require high-end hardware specifications. It can run in any device that support web services.

### Platform

KeyDash is developed as a web-based application, built using modern web technologies to ensure cross-platform compatibility and seamless accessibility. Unlike native mobile apps, it runs directly in web browsers, eliminating the need for downloads or installations while providing full functionality across desktops, laptops, and tablets. The application leverages React.js for its responsive frontend and Supabase for backend services, delivering real-time typing analytics, user authentication, and cloud-based data synchronization. As a web app, KeyDash prioritizes universal access, allowing users to practice typing anytime, anywhere, without device restrictions—simply by visiting the platform through a browser. This approach combines the flexibility of web applications with the performance of native-like experiences, making it an ideal solution for typing enthusiasts, students, and professionals alike.

## Technology Used

### React Technology

React is an open-source JavaScript library developed by Facebook for building dynamic and responsive user interfaces. In the development of KeyDash, React was chosen as the primary frontend technology due to its component-based architecture, efficient rendering with the Virtual DOM, and strong ecosystem of tools and libraries.

React uses JavaScript (JSX) for declarative UI development, allowing seamless integration of HTML-like syntax within JavaScript. This approach enables developers to create reusable UI components, making the codebase modular and easier to maintain. For KeyDash, this meant efficiently rendering real-time typing metrics, such as WPM (Words Per Minute) and accuracy, with smooth updates.

One of React most powerful features is state management, which allows dynamic data handling without full page reloads. Using React Hooks (useState, useEffect), KeyDash tracks user input, calculates performance metrics, and updates the UI instantly. Additionally, React unidirectional data flow ensures predictable behavior, making debugging and testing more straightforward.

For styling, CSS-in-JS libraries (such as Tailwind CSS) were used to create a responsive and visually appealing interface. React compatibility with Supabase also enabled seamless integration for user authentication and data storage, allowing users to save and track their typing progress over time.

Overall, React proved to be an excellent choice for KeyDash, offering high performance, cross-platform compatibility (web and mobile browsers), and a streamlined development process making it ideal for a real-time, interactive typing speed tester.

## Programming Language Used

### JavaScript Programming Language

JavaScript is a versatile, high-level programming language that serves as the core technology for web development. As the primary language for KeyDash, JavaScript enables dynamic and interactive web functionality, making it ideal for real-time applications like typing speed tests.

JavaScript operates within the browser's runtime environment, allowing immediate execution of code without compilation. Its event-driven architecture supports asynchronous operations, which is crucial for KeyDash to handle user inputs, calculate typing metrics (WPM, accuracy), and update the UI seamlessly.

KeyDash harnesses several powerful JavaScript features to deliver optimal performance, including DOM manipulation for real-time webpage updates that display typing results instantly, seamless React integration that enables component-based UI development through JavaScript's foundational support, and robust JSON handling capabilities that facilitate smooth data processing for Supabase interactions and local storage management.

## SQL (Structured Query Language)

SQL is the standard language for managing relational databases. In KeyDash, SQL powers data operations through Supabase, a PostgreSQL-based platform that offers full SQL support while adding modern features like real-time subscriptions and auto-generated APIs.

KeyDash leverages core SQL operations through Supabase PostgreSQL backend, including data retrieval of user history (such as previous WPM scores) using SELECT queries, profile and test result modifications via UPDATE commands, and performance trend analysis utilizing PostgreSQL's powerful analytical functions for comprehensive typing insights.

Unlike NoSQL alternatives, Supabase leverages PostgreSQL's strict SQL compliance, enabling complex queries with joins, transactions, and referential integrity – crucial for features like leaderboards and progress tracking.

## Software Tools Used

### Visual Studio Code

Visual Studio Code (VS Code) is a lightweight yet powerful source code editor developed by Microsoft, used as the primary development environment for KeyDash. Its built-in support for JavaScript, React, and SQL (via extensions) made it ideal for implementing the typing tester's frontend logic and database interactions. Features like IntelliSense for code completion, integrated terminal, and Git version control streamlined the development process, while extensions like ESLint and Prettier maintained code quality. The debugger tool was particularly useful for testing real-time typing analytics before deployment.

### Git and GitHub

Git served as the version control system for tracking all code changes in KeyDash, while GitHub hosted the repository remotely. This combination enabled seamless collaboration—branching prevented conflicts during feature development (e.g., Supabase integration), and pull requests ensured code reviews before merging. GitHub

Issues helped organize tasks like UI improvements and bug fixes, facilitated demo deployments during testing phases.

## Web browser

Modern browsers like Chrome and Edge were essential for testing KeyDash's responsiveness and real-time features. Their DevTools allowed debugging JavaScript performance, inspecting DOM updates during typing tests, and auditing Lighthouse metrics for optimization (e.g., WPM calculation speed). Browser extensions like React Developer Tools further simplified UI component inspection.

## Adobe Illustrator

Adobe Illustrator was used to design KeyDash's logo and custom UI assets (e.g., animated progress indicators). Its vector-based workflow ensured graphics remained sharp across devices, while SVG exports integrated smoothly with React for dynamic theming.

## Vercel

My React + Vite website was deployed using Vercel, a technology renowned for its ease of usage and performance. Every time we upload improvements to GitHub, Vercel builds and launches our website immediately. We were able to test features using live preview URLs and iterate more quickly thanks to our continuous deployment strategy. Only the build command, output directory, and required .env file needed to be specified for Vercel's smooth support of Vite.

## Coding Style Followed

KeyDash follows a component-based architecture using React, which promotes modularity and reusability in the codebase. The UI is broken down into independent components each managing its own state and logic. This approach ensures clean separation of concerns and simplifies debugging. Event-driven programming handles real-time user interactions, such as keystroke detection and WPM calculation, with React hooks managing state updates efficiently.

For backend interactions with Supabase, functional programming principles are applied, emphasizing pure functions and predictable data flow. SQL queries are structured in dedicated service modules, keeping database logic separate from UI components. Consistent code formatting enforces readability, while descriptive variable names (e.g., `wpm` instead of `x`). This hybrid style combining React's declarative UI patterns with functional data handling ensures a scalable and maintainable typing tester.

# Testing

## Test Plan

Manual testing is inefficient and doesn't scale. Test designs are used for reuse which improve our application efficiency. So, we have done unit test, integration testing, black-box and white-box testing and system testing as a whole.

## Test Cases

### Testing Cases Summary

By looking at the characteristics of developed smart phone application, it tries to understand the factors that lead to a note uploading system, but also figures out the problem that may lead to the failure we have done different test our application project and the result of upgrading the application for future prospects to improve the application.

S.N.	Test Case Summary	Results
1	To test whether it displays differently to incorrect character.	Passed
2	To test whether it displays real time WPM meter.	Passed
3	To test whether it displays score and graph.	Passed
4	To test whether it displays leaderboard.	Passed
5	To test whether it keeps the record of the user progress or not.	Passed
6	To test whether it displays error page in case of invalid URL.	Passed

Table 7.1: Testing Cases Summary

## Sample Test Cases

<b>Test Cases</b>	<b>1</b>
Test Objective	To test whether it displays differently to incorrect character.
Test Data	tn the end, we wokl remember rao the words of our enemies.
Expected Result	Error should have a font color of red with character being underlined.
Actual Result	The error is underlined and it's font color is red.
Conclusion	Work out is successful.

Table 7.2: Test Case 1

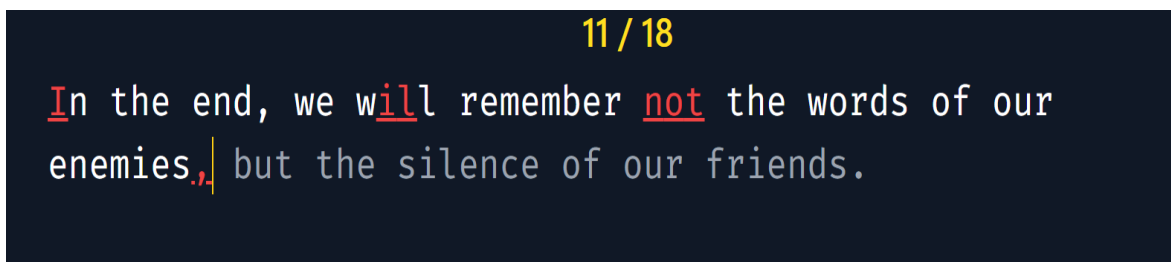


Figure 7.1: Test case 1

<b>Test Cases</b>	<b>2</b>
Test Objective	To test whether it displays real time metrics (such as WPM, error accuracy).
Test Data	All he ever wanted out of life wad lave. You see, he just didn't have any to give
Expected Result	The all 3 metrics (WPM, Accuracy and error) should be displayed correctly according to the data.



Actual Result	The key metrics data is accurately shown.
Conclusion	Work out is successful.

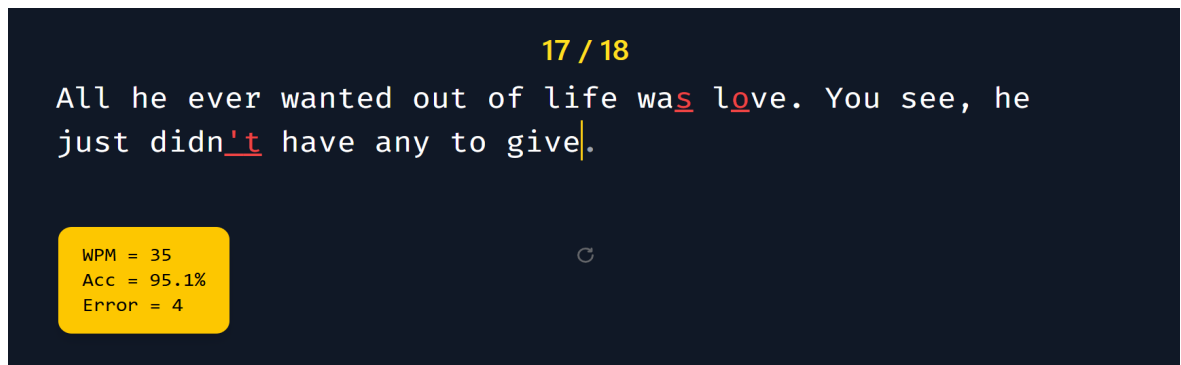


Figure 7.2: Test case 2

<b>Test Cases</b>	<b>3</b>
Test Objective	To test whether it displays score and graph.
Expected Result	After completion of session all of key metrics (WPM, accuracy, mistakes, characters, time taken) with the graphical representation of WPM and mistakes should be displayed.
Actual Result	All key metrics with graph is displayed.
Conclusion	Work out is successful.

Table 7.4: Test Case 3

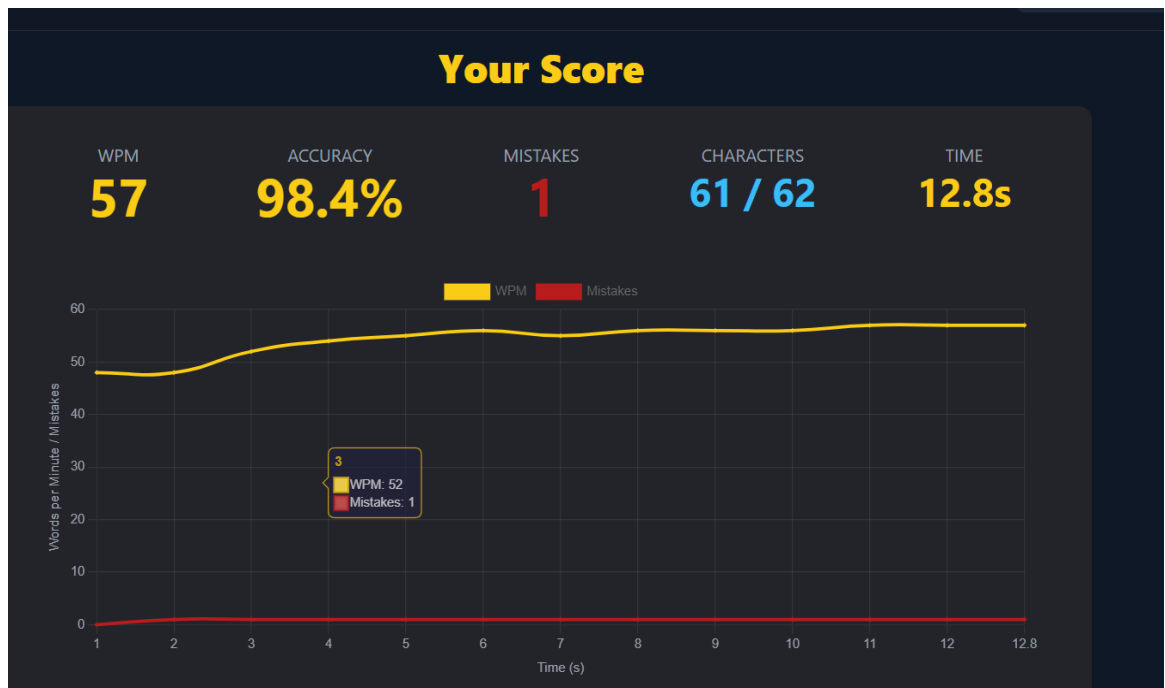
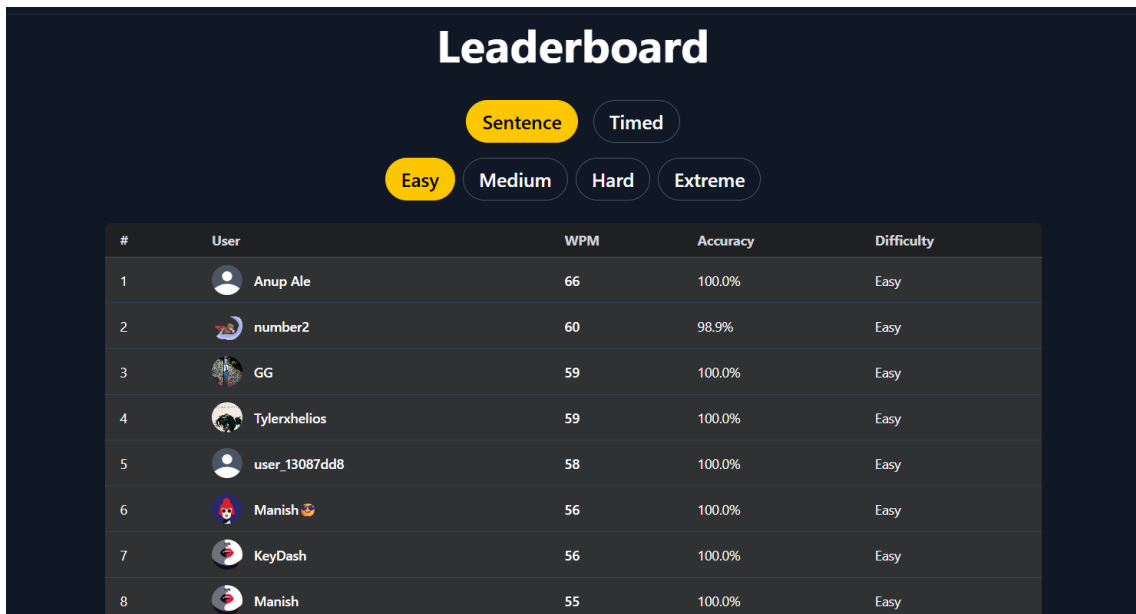


Figure 7.3: Test case 3

Test Cases	4
Test Objective	To test whether it displays leaderboard.
Test Data	Leaderboard of sentence mode with easy difficulty is selected.
Expected Result	All of the user participated in the selected mode's high score is displayed and ranked in table according to their WPM.
Actual Result	Leaderboard of easy sentence mode is displayed with user ranking according to their WPM.
Conclusion	Work out is successful.

Table 7.5: Test Case 4



The screenshot shows a 'Leaderboard' interface with a dark background. At the top, there are two rows of difficulty and mode buttons. The first row has 'Sentence' (highlighted in yellow) and 'Timed'. The second row has 'Easy' (highlighted in yellow), 'Medium', 'Hard', and 'Extreme'. Below these buttons is a table with 5 columns: '#', 'User', 'WPM', 'Accuracy', and 'Difficulty'. The table lists 8 users ranked by WPM.

#	User	WPM	Accuracy	Difficulty
1	Anup Ale	66	100.0%	Easy
2	number2	60	98.9%	Easy
3	GG	59	100.0%	Easy
4	Tylerxhelios	59	100.0%	Easy
5	user_13087dd8	58	100.0%	Easy
6	Manish 🍌	56	100.0%	Easy
7	KeyDash	56	100.0%	Easy
8	Manish	55	100.0%	Easy

Figure 7.4: Test case 4

Test Cases	5
Test Objective	To test whether it keeps the record of the user progress or not.
Test Data	Open user profile.
Expected Result	The metrics (WPM and accuracy) with user past session record should be displayed.
Actual Result	The detailed record of past session and user highest WPM and average accuracy is displayed.
Conclusion	Work out is successful.

Table 7.6: Test Case 5

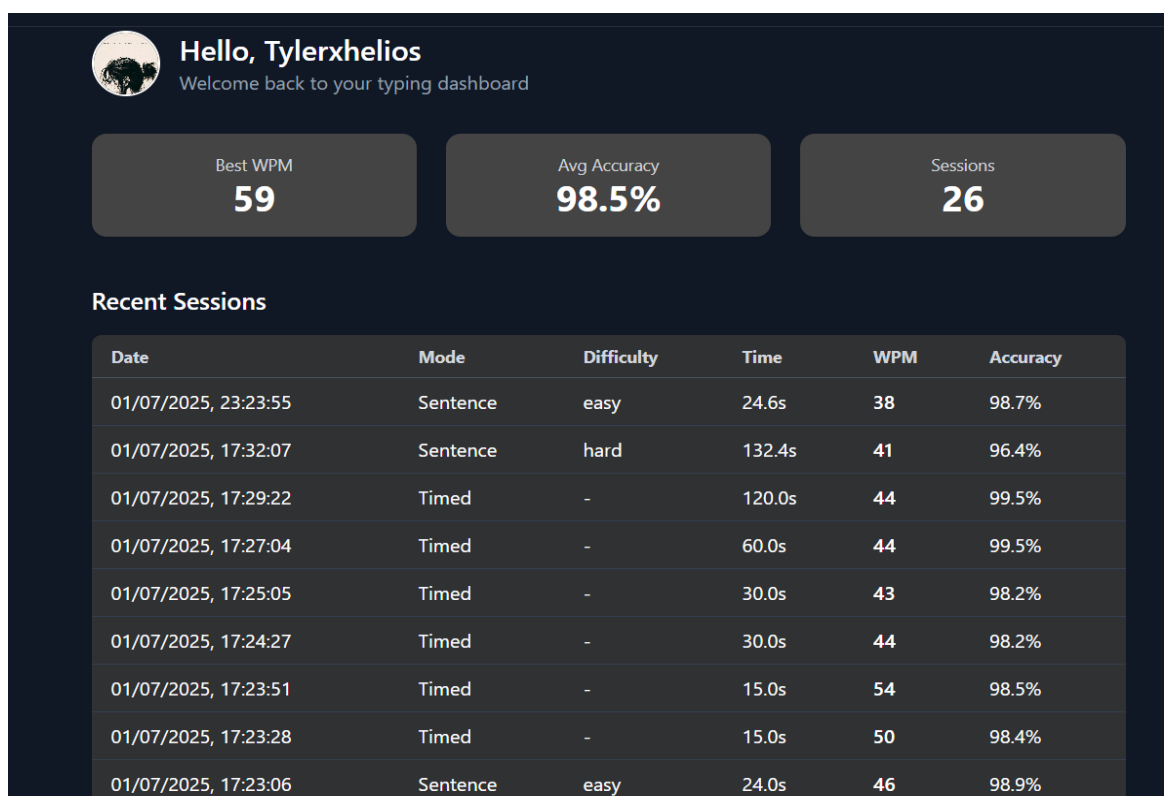


Figure 7.5: Test case 5

Test Cases	6
Test Objective	To test whether it displays error page in case of invalid URL.
Test Data	https://keydash.shresthamanish.info.np/kdjfskdjff
Expected Result	Since the URL is incorrect or invalid the error page should be loaded.
Actual Result	The typed URL being invalid, error page is displayed.
Conclusion	Work out is successful.

Table 7.7: Test Case 6

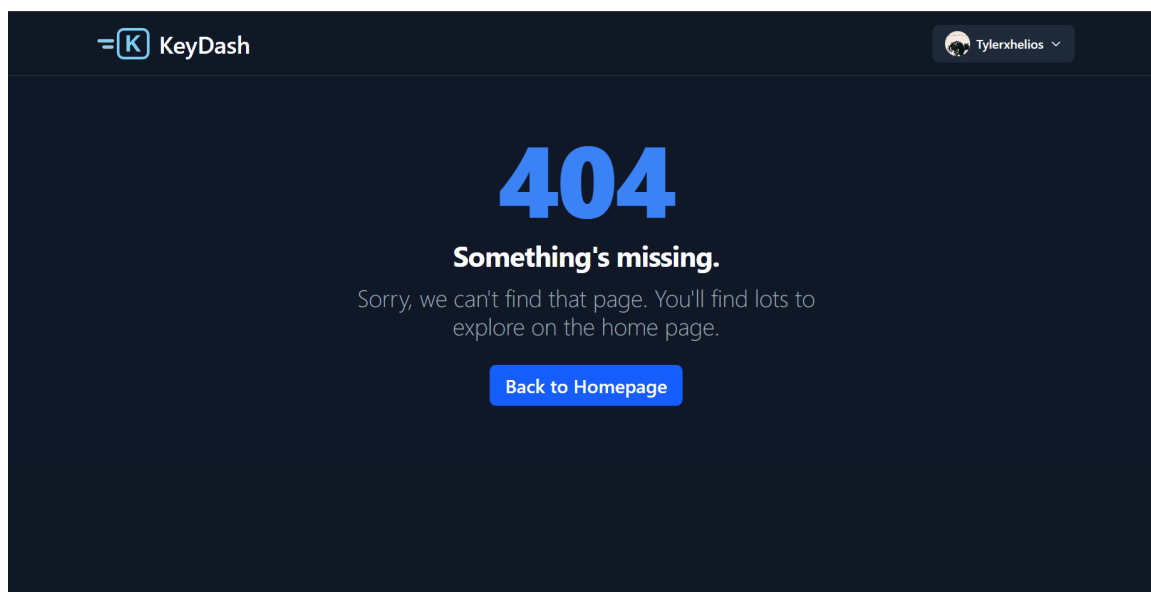


Figure 7.6: Test case 6

# Limitations And Recommendations

## Limitations

However, we attempt to banish all of our encountered errors. Some limitations are:

- KeyDash requires constant internet connectivity for Supabase integration, lacking offline functionality.
- The tool provides only basic metrics (WPM/accuracy) without advanced typing analytics or personalized feedback.
- No real-time multiplayer mode exists for competitive typing races.
- Mobile testing accuracy may be skewed by on-screen keyboard auto-correction features.
- Users cannot practice with custom text (e.g., code, non-English content) due to fixed test passages.

## Recommendation

These limitations can be solved by further enhancement of the system. We can further add the advanced searching features and also use AI for the recommendation. We can further make more attractive UI in the future as per the recommendations by the users.

# Conclusion

## Conclusion

KeyDash successfully delivers an efficient and user-friendly typing speed tester that helps users measure and improve their typing proficiency through real-time analytics. By leveraging modern web technologies like React.js and Supabase, the project achieves a responsive, interactive interface with accurate WPM and accuracy calculations. The integration of Supabase ensures secure data storage and retrieval, while the clean UI design enhances user experience across devices. Although the current version has some limitations, such as lack of offline functionality and advanced analytics, it provides a solid foundation for future enhancements.

Overall, KeyDash demonstrates the effective application of full-stack development principles to create a functional and scalable typing assessment tool. The project not only fulfills its core objective of evaluating typing speed but also showcases the potential for expansion with features like multiplayer mode, customized practice texts, and detailed performance insights. As a college project, it highlights the developer's technical competency in building web applications while leaving room for iterative improvements to meet diverse user needs.

We thank you all who have helped us directly or indirectly.

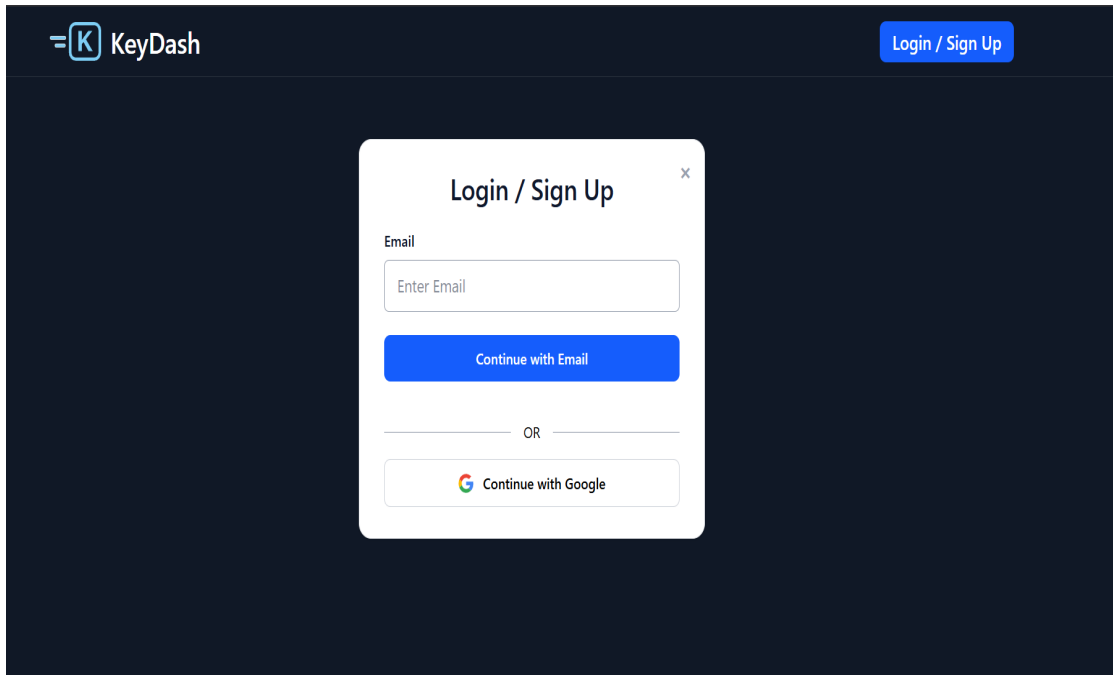
We have tried our best to give you the very best result.

## REFERENCES

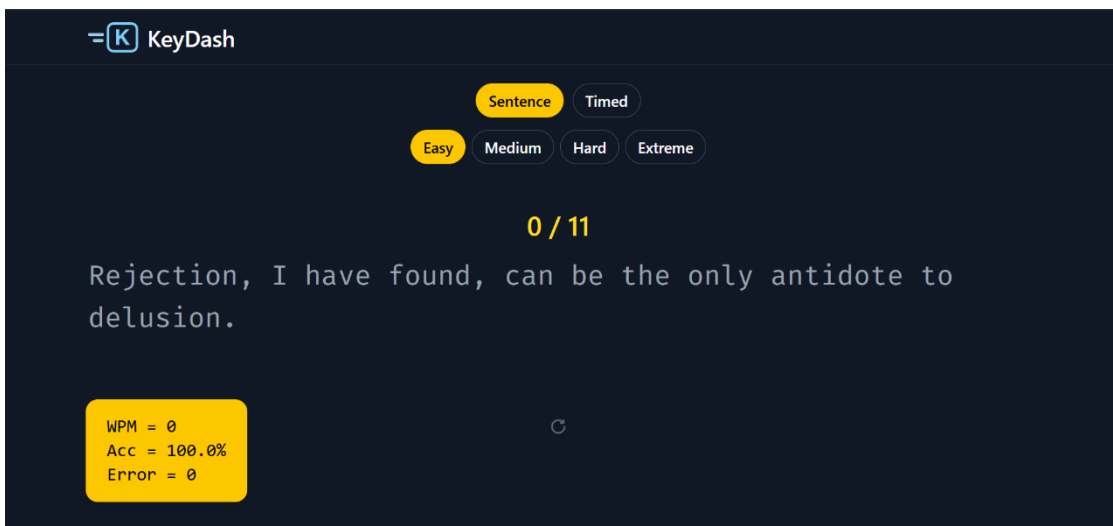
- [1] D. H. Smith and J. A. Doe, "The Impact of Real-Time Feedback on Typing Proficiency," *Journal of Computer-Assisted Learning*, vol. 12, no. 3, pp. 45–60, 2020.
- [2] R. Brown and M. Lee, "Cognitive Load Theory and Skill Acquisition in Typing," *International Journal of Human-Computer Studies*, vol. 78, pp. 112–125, 2021.
- [3] 10FastFingers, "Typing Speed Test Methodology," 2019. [Online]. Available: <https://www.10fastfingers.com/methodology>
- [4] TypingClub, "Gamified Learning in Typing Education," *EdTech Review*, 2022. [Online]. Available: <https://www.typingclub.com/research>
- [5] A. Taylor et al., "Engagement in Digital Typing Tutors: A Meta-Analysis," *Computers & Education*, vol. 95, pp. 89–102, 2021.
- [6] K. Patel and S. Kim, "Limitations of Current Typing Tools in Structured Learning," *Proc. of the ACM Conference on Learning Systems*, pp. 205–220, 2022.
- [7] L. Garcia, "Adaptive Learning Systems for Typing Mastery," *IEEE Transactions on Education*, vol. 65, no. 4, pp. 550–565, 2023.



## APPENDICES



### Appendix A: Login Screen



### Appendix B: Sentence mode



## Appendix C: Timed mode

KeyDash

Tylerxhelios

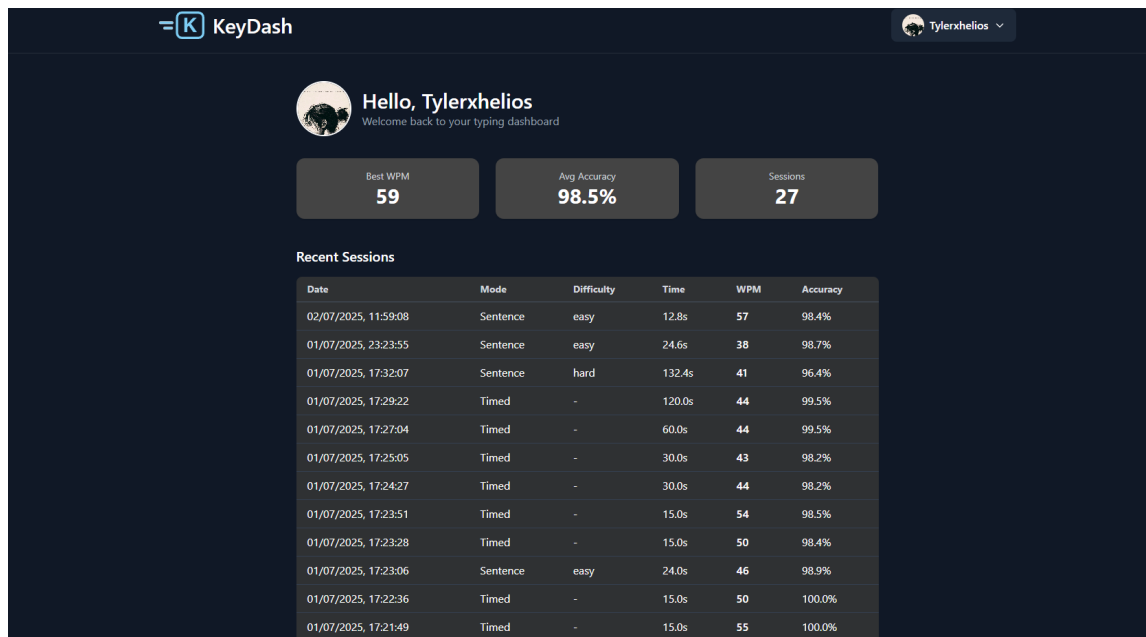
Leaderboard

Sentence Timed

Easy Medium Hard Extreme

#	User	WPM	Accuracy	Difficulty
1	Milan	75	100.0%	Easy
2	Anup Ale	66	100.0%	Easy
3	number2	60	98.9%	Easy
4	GG	59	100.0%	Easy
5	Tylerxhelios	59	100.0%	Easy
6	user_13087dd8	58	100.0%	Easy
7	Manish	56	100.0%	Easy
8	KeyDash	56	100.0%	Easy
9	Manish	55	100.0%	Easy
10	Manish Shrestha	53	100.0%	Easy


## Appendix D: Leaderboard



## Appendix E: Profile Section

## Account Settings

Manage your account.



Upload profile picture

Display Name \*

Tylenxhelios

Email

anuragmgr06@gmail.com

Bio

Write a few sentences about yourself.

Website URL

https://www.example.com

X URL

https://www.x.com/yourhandle

Instagram URL

https://www.instagram.com/yourhan

Github URL

https://www.github.com/yourhandle

Youtube Channel URL

https://www.youtube.com/yourchann

LinkedIn URL

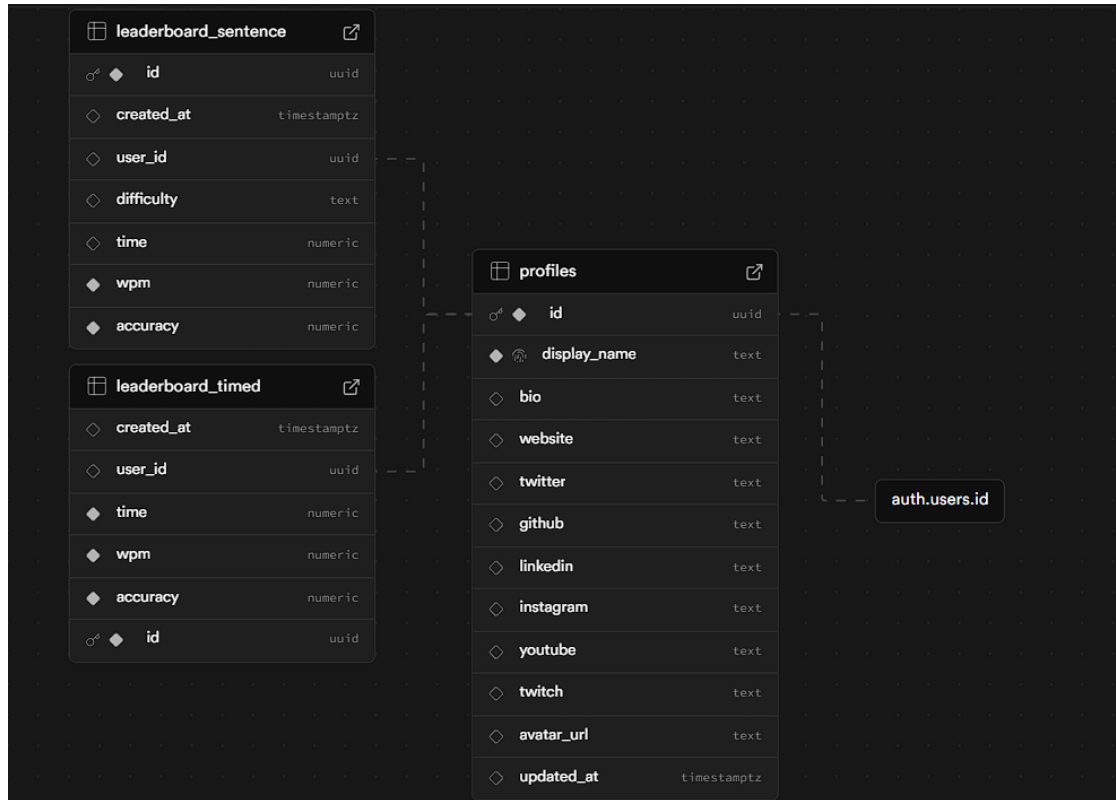
https://www.linkedin.com/in/yourhan

Twitch URL

https://www.twitch.tv/yourchannel

Update Profile

## Appendix F: Profile Update Setting



## Appendix G: Backend Database