Testing documentation

1. Testing functional requirements

1) User Registration and Profile Management:

- Test 1.1:
  - **Precondition:** User is on the registration page.
  - **Test Description:** Verify that a user can create an account with valid credentials via the registration endpoint.
  - **Postcondition:** User account is created and stored in the database.
- Test 1.2:
  - **Precondition:** User attempts to register with invalid credentials.
  - **Test Description:** Verify that error messages are displayed for invalid registration inputs (e.g., missing fields, weak password) via the registration endpoint.
  - **Postcondition:** Error messages are displayed, and the account is not created.
- Test 1.3:
  - **Precondition:** User has an existing account and is on the login page.
  - **Test Description:** Verify that a user can log in with correct credentials via the login endpoint.
  - **Postcondition:** User is authenticated and redirected to the dashboard.
- Test 1.4:
  - **Precondition:** User attempts to log in with incorrect credentials.
  - **Test Description:** Verify that a user cannot log in with incorrect credentials via the login endpoint.
  - **Postcondition:** Error message is displayed, and the user remains on the login page.
- Test 1.5:
  - **Precondition:** User is logged in and on the profile management page.
  - **Test Description:** Verify that a user can update personal information in their profile via the profile update endpoint.
  - **Postcondition:** User's profile information is updated in the database.
- Test 1.6:
  - **Precondition:** User is logged in and on the profile page.
  - **Test Description:** Verify that a user can track progress and manage preferences via the profile endpoint.
  - **Postcondition:** User's progress and preferences are displayed correctly.

2) Recording/Sample Selection:

- Test 2.1:
  - **Precondition:** User is logged in and on the song selection page.
  - **Test Description:** Verify that a user can choose a selected recording from his device.
  - **Postcondition:** List of the available recordings is displayed.
- Test 2.2:
  - **Precondition:** User is logged in and on the voice sample upload page.
  - **Test Description:** Verify that a user can upload their own voice samples via the voice sample upload endpoint.
  - **Postcondition:** Voice sample is uploaded and stored in the database.
- Test 2.3:
  - **Precondition:** User is logged in and on the difficulty level selection page.
  - **Test Description:** Verify that the app supports various difficulty levels for songs via the difficulty level endpoint.
  - **Postcondition:** List of available difficulty levels is displayed.

3) Real-time Audio Feedback:

- Test 3.1:
  - **Precondition:** User is performing a song.
  - **Test Description:** Verify that the app provides real-time feedback on pitch and words accuracy via the feedback endpoint.
  - **Postcondition:** Real-time pitch accuracy feedback is displayed to the user.
- Test 3.2:
  - **Precondition:** User is performing a song.
  - **Test Description:** Verify that the app provides real-time feedback on timing via the feedback endpoint.
  - **Postcondition:** Real-time timing feedback is displayed to the user.

4) Voice Comparison and Analysis:

- Test 4.1:
  - **Precondition:** User has completed a performance and now waiting for additional feedback.
  - **Test Description:** Verify that the app provides detailed analysis of pitch, tone, and rhythm via the analysis endpoint.
  - **Postcondition**: Detailed analysis is generated and displayed.
- Test 4.2:
  - **Precondition:** User has completed a performance and now waiting for additional feedback.
  - **Test Description:** Verify that the app highlights areas for improvement via the analysis endpoint.
  - **Postcondition:** Areas for improvement are highlighted and displayed.

5) Visualization of Performance:

- Test 5.1:
  - **Precondition:** User has completed a performance.
  - **Test Description:** Verify that the app generates the necessary graphs for user performance via the visualization endpoint.
  - **Postcondition:** the necessary graphs are generated and displayed.
- Test 5.2:
  - **Precondition:** User has completed a performance.
  - **Test Description:** Verify that the app generates all the relevant visual feedback metrics (depends on the consumer feedback) via the visualization endpoint.
  - **Postcondition:** Relevant visual feedback metrics are generated and displayed.

6) Scoring and Progress Tracking:

- Test 6.1:
  - **Precondition:** User has completed a performance.
  - **Test Description:** Verify that the app assigns scores based on various performance criteria via the relevant endpoint.
  - **Postcondition:** Scores are generated and displayed.
- Test 6.2:
  - **Precondition:** User is logged in and on the progress tracking page.
  - **Test Description:** Verify that the app tracks user progress over time via the progress tracking endpoint.
  - **Postcondition:** User's progress over time is displayed.
- Test 6.3:
  - **Precondition:** User is logged in and on the performance history page.
  - **Test Description:** Verify that users can view their performance history and improvements via the progress tracking endpoint.
  - **Postcondition:** User's performance history and improvements are displayed.

7) Customization of Practice Sessions:

- Test 7.1:
  - **Precondition**: User is logged in and on the practice session page.
  - **Test Description:** Verify that a user can adjust playback speed during practice sessions via the customization endpoint.
  - **Postcondition:** Playback speed is adjusted for the practice session.
- Test 7.2:
  - **Precondition:** User is logged in and on the practice session page.
  - **Test Description:** Verify that a user can adjust tempo and rhythm for practice sessions via the customization endpoint.
  - **Postcondition:** tempo and rhythm are adjusted for the practice session.

- Test 7.3:
    - **Precondition:** User is logged in and on the practice session customization page.
    - **Test Description:** Verify that a user can select different difficulty levels for practice sessions via the customization endpoint.
    - **Postcondition:** Difficulty level is set for the practice session.

8) Data Security and Privacy:

- Test 8.1:
    - **Precondition:** User is attempting to log in.
    - **Test Description:** Verify using authentication mechanisms that user is shown the relevant data and pages for him via the authentication endpoint.
    - **Postcondition:** Secure authentication is confirmed, and user is logged in.
- Test 8.2:
    - **Precondition:** User data is being accessed.
    - **Test Description:** Verify that sensitive information is protected from unauthorized access using the get data endpoints.
    - **Postcondition:** Sensitive information is confirmed to be protected.

9) Teacher and Student Interactions:

- Test 9.1:
    - **Precondition:** User is logged in and on the class creation page.
    - **Test Description:** Verify that a user can create a class via the class creation endpoint.
    - **Postcondition:** Class is created and stored in the database.
- Test 9.2:
    - **Precondition:** Student is logged in and enrolled in a class.
    - Test Description: Verify that students can access tasks within their enrolled classes via the task's endpoint.
    - **Postcondition:** List of tasks is displayed for the student.
- Test 9.3:
    - **Precondition:** Student is logged in and on the performance recording page.
    - **Test Description:** Verify that students can record their performances and receive feedback via the performance recording endpoint.
    - **Postcondition:** Performance is recorded, and feedback is available.
- Test 9.4:
    - **Precondition:** Teacher is logged in and on the student performance review page.
    - **Test Description:** Verify that teachers can review student performances via the review endpoint.
    - **Postcondition:** Student performances are available for review.
- Test 9.5:

    - **Precondition:** Teacher is logged in and reviewing a student performance.
    - **Test Description:** Verify that teachers can provide feedback to students via the feedback endpoint.
    - **Postcondition:** Feedback is submitted and available to the student.
- Test 9.6:
    - **Precondition:** Teacher and student are logged in and communicating.
    - **Test Description:** Verify that teachers and students can engage in communication regarding progress and questions via the chat endpoint.
    - **Postcondition:** Communication is stored and accessible.
- Test 9.7:
    - **Precondition:** Student is logged in and, on the feedback, viewing page.
    - **Test Description:** Verify that students can view teacher feedback within the app via the feedback viewing endpoint.
    - **Postcondition:** Teacher feedback is displayed to the student.
- Test 9.8:
    - **Precondition:** Student is logged in and on the question submission page.
    - **Test Description:** Verify that students can ask questions and seek clarification from teachers via the question endpoint.
    - **Postcondition:** Questions are submitted and available to the teacher.

2 Testing non-functional requirements

Performance Constraints:

1. **Throughput:**
    - **Testing Method:** Load Testing
    - **Description:** Simulate 100 concurrent users accessing and using the system's features to ensure the system can handle the load without performance degradation.
    - **Tools we can Use:** Apache JMeter
2. **Real-time:**
    - **Testing Method:** Performance Testing
    - **Description:** Test the system's ability to provide real-time feedback during coaching sessions. Measure the response time and ensure it remains within acceptable limits.
    - **Tools we can Use:** Locust
3. **Speed:**
    - **Testing Method:** Response Time Testing
    - **Description:** Measure the response time for various user actions (registration, login, logout, etc.) to ensure they are handled within 5 seconds.
    - **Tools we can Use:** Selenium WebDriver, Postman
4. **CPU Usage:**
    - **Testing Method:** Resource Utilization Monitoring
    - **Description:** Monitor the CPU usage of the system to ensure it does not exceed 20% of the device's CPU.
    - **Tools we can Use:** New Relic, Datadog
5. **Memory:**
    - **Testing Method:** Memory Usage Testing
    - **Description:** Ensure the app size is no more than 5MB and monitor memory usage to ensure efficient memory management.
    - **Tools we can Use:** Tools in React Native

Reliability & Stability:

1. **Fault Tolerance:**
    - **Testing Method:** Stress Testing, Recovery Testing
    - **Description:** Simulate unexpected failures (e.g., loss of internet connection, dead battery) to ensure the system can recover gracefully.
    - **Tools we can Use:** Chaos Monkey, Custom Scripts
2. **Data Integrity:**
    - **Testing Method:** Data Validation Testing
    - **Description:** Ensure that data is handled accurately, and recordings are not damaged.
    - **Tools we can Use:** Custom Test Scripts
3. **Error Handling:**
    - **Testing Method:** Error Handling Testing
    - **Description:** Verify that informative error messages and logs are provided for troubleshooting and debugging.
    - **Tools we can Use:** Custom Test Scripts, Log Analysis Tools

4. **Back-Up:**
    - **Testing Method:** Backup and Restore Testing
    - **Description:** Ensure the system performs daily database backups and can restore data from backups.
    - **Tools we can Use:** AWS Backup Services, Custom Scripts
5. **User-History:**
    - **Testing Method:** Logging Testing
    - **Description:** Ensure that user actions are logged for backtrace purposes.
    - **Tools we can Use:** ELK Stack Custom Scripts.

Safety & Security:

1. **Discrete:**
    - **Testing Method:** Access Control Testing
    - **Description:** Ensure that user recordings are not accessible to unauthorized individuals.
    - **Tools we can Use:** Custom Test Scripts
2. **User Authenticator:**
    - **Testing Method:** Authentication Testing
    - **Description:** Verify that the system authenticates user identity during login and profile actions.
    - **Tools we can Use:** OWASP ZAP, Postman
3. **Encryption:**
    - **Testing Method:** Security Testing
    - **Description:** Ensure sensitive information is encrypted using SHA-512.
    - **Tools we can Use:** OWASP ZAP, SSL Labs
4. **Access Control:**
    - **Testing Method:** Role-Based Access Control Testing
    - **Description:** Verify that different user roles have distinct access levels to system functionalities.
    - **Tools we can Use:** Custom Test Scripts

Portability:

1. **OS Compatibility:**
    - **Testing Method:** Cross-Platform Testing
    - **Description:** Ensure the system is compatible with popular OS systems (Android, iOS, Microsoft).
    - **Tools we can Use:** Appium, BrowserStack
2. **Multilingual Support:**
    - **Testing Method:** Language testing
    - **Description:** Verify that the system supports text in English and Hebrew, utilizing Unicode for character representation.
    - **Tools we can Use:** Custom Test Scripts

Usability:

1. **User Training:**
    - **Testing Method:** Usability Testing

- **Description:** Verify that the system is easy to use and provides necessary aids (e.g., detailed icons).
- **Tools we can Use:** Custom Test Scripts, User Feedback

Availability:

1. **System Availability:**
   - **Testing Method:** Uptime Monitoring
   - **Description:** Ensure the system is available to users 99% of the time (1% for scheduled maintenance).
   - **Tools we can Use:** Pingdom, New Relic

Platform Constraints Testing

1. **Real-time Voice Recognition:**
   - **Testing Method:** Real-Time Performance Testing
   - **Description:** Ensure the app can analyze and present real-time feedback.
   - **Tools we can Use:** Custom Test Scripts
2. **Database Storage for Audio Files:**
   - **Testing Method:** Database Performance Testing
   - **Description:** Ensure the database efficiently manages and preserves user-generated audio content.
   - **Tools we can Use:** PostgreSQL Benchmarking Tools
3. **Cross-Platform Compatibility:**
   - **Testing Method:** Cross-Platform Testing
   - **Description:** Ensure the platform is universally compatible across iOS, Android, and web.
   - **Tools we can Use:** Appium, BrowserStack

Special Restrictions and Limitations Testing

1. **Microphone Quality and Environment:**
   - **Testing Method:** Environmental Testing
   - **Description:** Advise users to use a high-quality microphone for accurate voice recognition.
   - **Tools we can Use:** Custom Test Scripts
2. **Multi-Language Support:**
   - **Testing Method:** Language Testing
   - **Description:** Ensure the system supports multiple languages, focusing on Hebrew and Torah reading.
   - **Tools we can Use:** Custom Test Scripts

3 Test-Driven Development:

In the development of our app, we did not utilize the Test-Driven Development (TDD) strategy. The primary reason for this was the need to research and evaluate various tools and technologies to determine the most suitable ones for our project. This exploratory phase made it impractical to write tests prior to the actual code development.

4 Random & automatically generated tests

In the development of our application, we did not implement random or automatically generated tests. The primary reason for this was our focus on developing and researching for the beta version of the app. Given the time constraints and the need to prioritize core functionality and tool validation, we were unable to allocate sufficient time and resources to write and integrate random or automatically generated tests into our testing strategy.

5 Testing the user interface

UI Testing Methods:

1. **Manual Testing:**
    - **Description:** We thoroughly tested the user interface by manually using the application extensively. This hands-on approach allowed us to interact with each component and verify its functionality under various conditions.
    - **Device Diversity:** To ensure broad compatibility and a seamless user experience, we tested the UI on multiple devices with different operating systems, including both Android and iOS. This helped us identify and resolve device-specific issues.
2. **Cross-Platform Testing:**
    - **Description:** By testing on a variety of devices and screen sizes, we ensured that the UI elements were responsive and displayed correctly across different environments. This was crucial for maintaining a consistent user experience on both smartphones and tablets.

We have not yet implemented online help or context-sensitive assistance. As we progress beyond the beta phase, we plan to explore automated UI testing and check tools such as:

- **Selenium:** For automating web application testing.
- **Appium:** For automating mobile application testing on both Android and iOS.
- **React Native Testing Library:** For unit and integration testing of React Native components.
- **BrowserStack or Sauce Labs:** For cross-browser and cross-device testing to ensure consistent behavior across various platforms.

6 Testing build, integration & deployment

Clean Build Verification:

We ensured that our application builds cleanly and integrates well with its components and outside modules through the following measures:

1. **Docker for Backend:**
    - **Description:** We built the backend using Docker, which containerizes the application and its dependencies. This approach ensures a consistent environment across different development.
    - **Integration:** Docker allows seamless integration with relevant servers and services, minimizing compatibility issues.
2. **React Native for Frontend:**
    - **Description:** We chose React Native for the frontend to ensure compatibility with both Android and iOS systems. This cross-platform framework simplifies the development process and maintains a consistent user experience across different devices.
3. **AWS EC2 for Server Deployment:**
    - **Description:** Our application is deployed on an AWS EC2 instance. We have configurations and scripts in place to set up, test, and maintain the server.
    - **Continuous Integration:** We utilize continuous integration (CI) tools like GitHub Actions to automate the build and integration process, ensuring that new code changes do not break the application.

Installation Measures:

1. **Application Installation Verification:**
    - **Configuration Scripts:** We use configuration scripts to set up the application environment on the server. These scripts ensure that all necessary dependencies and configurations are correctly applied.
    - **Installation Tests:** After deploying the application, we run a series of manual tests to verify that the application is installed properly, and all services are running as expected. This includes checking database connections, API endpoints, and user interface functionality.
2. **Application Uninstallation Verification:**
    - **Uninstallation Scripts:** We have developed scripts to automate the uninstallation process. These scripts ensure that all components of the application are removed cleanly without leaving residual files or configurations.

Maintenance and Testing:

- **Server Maintenance:** We have commands and scripts to take the server down for maintenance and bring it back up. This ensures minimal downtime and a smooth transition during updates or fixes.
- **Regular Testing:** We regularly run manual tests to verify that all components of the application work well together. This includes testing interactions between the frontend and backend, as well as between the application and external modules.