Technical Design Document: DevType

Table of Contents

Introduction and System Overview

Introduction

DevType is a modern web application designed to test and improve typing speed. The platform offers users a seamless experience with features such as real-time typing tests, user authentication, personalized profiles, and performance tracking.

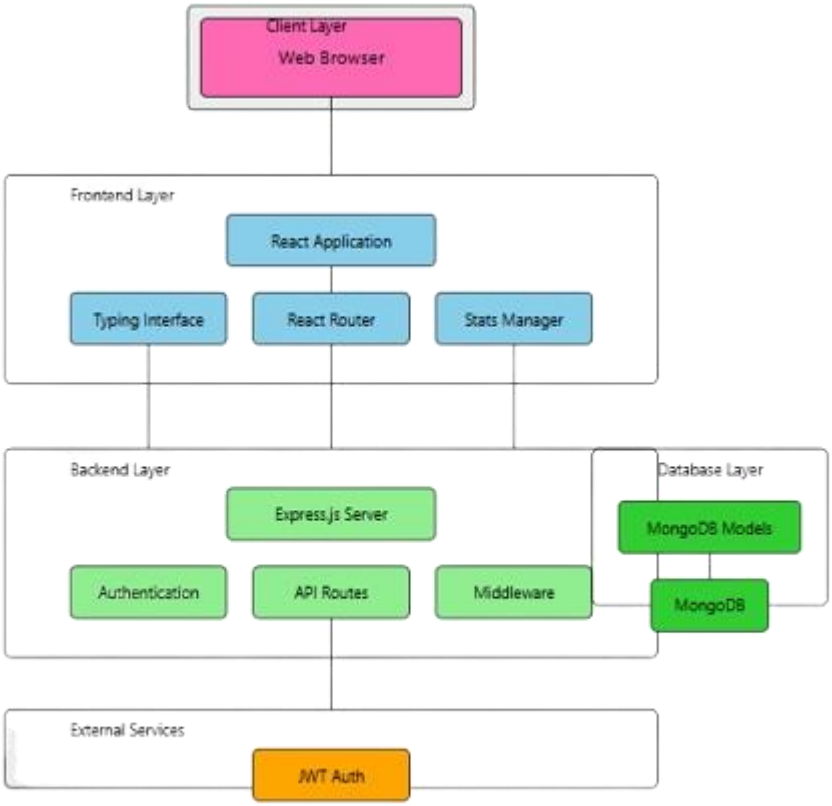
This document outlines the architecture, components, technology stack, and design decisions of DevType, serving as a guide for developers, stakeholders, and contributors.

System Overview

DevType is built using the MERN stack (MongoDB, Express.js, React, Node.js), leveraging modern web development practices. The application focuses on scalability, maintainability, and responsive design to provide an engaging experience across devices.

Architecture

High-Level Architecture



The system follows a three-tier architecture:

1. **Frontend:** Built with React.js to handle client-side user interfaces and interactions.
2. **Backend API:** Developed with Express.js and Node.js for server-side logic, API endpoints, authentication, and business logic.
3. **Database:** Uses MongoDB to store user data, typing test results, and related information.

Component Interactions

- **Frontend communicates with the Backend API** via RESTful endpoints for user data, test results, and authentication.
- **Backend interacts with the database** using Mongoose for CRUD operations.

Backend Design

Technologies Used

- **Node.js:** JavaScript runtime environment.
- **Express.js:** Web application framework for building APIs.
- **MongoDB:** NoSQL database for data storage.
- **Mongoose:** ODM library for MongoDB.
- **JWT:** JSON Web Tokens for authentication.
- **bcrypt:** Library for password hashing.

Project Structure

- **index.js:** Entry point of the server application.
- **middleware/:** Contains middleware functions, including authentication.
- **models/:** Mongoose schemas for User and TypingTest models.
- **routes/:** Defines API endpoints for authentication, users, and typing tests.

API Design

The backend exposes RESTful API endpoints categorized under:

- **Authentication (/api/auth):**
    - POST /login: User login and JWT token issuance.
    - POST /signup: User registration.
- **Users (/api/users):**
    - GET /:id: Retrieve user profile and typing test history.
    - PUT /:id: Update user profile.
- **Typing Tests (/api/tests):**
    - GET /: Fetch a list of typing tests.

    - POST /: Submit typing test results.

Database Schema

User Model (User.js):

- **Fields:**
    - name (String, required): User's full name.
    - email (String, required, unique): User's email address.
    - password (String, required): Hashed password.

Typing Test Model (TypingTest.js):

- **Fields:**
    - user (ObjectId, required): References the User model.
    - wpm (Number, required): Words per minute achieved.
    - accuracy (Number, required): Typing accuracy percentage.

Middleware:

Authentication Middleware (auth.js):

- Validates JWT tokens sent in the Authorization header.
- Attaches the authenticated user's information to the request object.
- Protects routes requiring authentication.

Frontend Design

Technologies Used

- **React.js:** JavaScript library for building user interfaces.
- **React Router DOM:** For client-side routing.
- **Tailwind CSS:** Utility-first CSS framework for styling.
- **Vite:** Build tool for faster development.

Project Structure

- **main.jsx:** Entry point of the React application.
- **App.jsx:** Main application component.
- **components/:** Reusable UI components.
- **pages/:** Page components corresponding to routes.
- **routes/:** Defines client-side routing.

- **lib/:** Utility functions.
- **index.css:** Global CSS and Tailwind directives.
- **public/:** Static assets.

Routing

Implemented using React Router:

- /: Homepage with typing test.
- /signup: User registration page.
- /login: User login page.
- /profile: User profile with test history.

State Management

- **Local State:** Managed using useState and useEffect hooks.
- **Authentication State:**
    - Accessed via custom hooks or utility functions.
- **Data Fetching:**
    - Utilizes fetch API.
    - Handles loading and error states.

Key Components

NavBar:

- Displays navigation links.
- Shows different options based on authentication state.
- Includes a logout mechanism.

Typing Test:

- Core component for the typing test.
- Displays the text to type and tracks user input.

Profile:

- Displays user information and test history.

Security Considerations

- **Authentication:**
    - JWT tokens securely generated and signed with a secret key.
    - Passwords hashed using bcrypt before storing in the database.
- **Authorization:**
    - Protected routes require valid JWT tokens.
    - Input validation to prevent injection attacks.
- **CORS Configuration:**
    - Configured to allow requests from trusted origins.

Deployment Plan

1. **Deployment:**
    - Build the React application using npm run build.
    - Host static files on services like Vercel or AWS.
2. **Database Hosting:**
    - Use managed MongoDB services like MongoDB Atlas.

Future Enhancements:

Technical Improvements:

- Add TypeScript for type safety.
- Implement Redux for complex state management.

Feature Enhancements:

- **Progress Tracking:** Allow users to track progress over time.
- **Custom Tests:** Enable users to create custom typing tests.
- **Real-Time Competition:** Allow users to compete with others in real-time.
- **Achievements:** Introduce badges and milestones for users.

Conclusion

DevType is a user-centric application designed to enhance typing skills through interactive tests and performance tracking. By utilizing modern technologies and best practices, DevType delivers a secure, scalable, and engaging platform for users.