Flashcard Extension Documentation

# Table of Contents

* 1. Overview
* 2. System Architecture
* - Frontend (React)
* - Backend (Express.js)
* 3. Key Components
* - Core Logic
* - Frontend Components
* - Backend Server
* - Gesture Recognition
* 4. Data Flow
* 5. API Reference
* 6. WebSocket Events
* 7. CSS Styling
* 8. Getting Started
* 9. Usage Guide
* 10. Development Notes
* 11. Extending the Application

# Overview

This application is a spaced repetition flashcard system with a React frontend and Express.js backend. It features a card scheduling algorithm, gesture-based card rating, and real-time updates via WebSockets. This extension should add flashcards and then by using gestures it should get the feedback.

# System Architecture

## Frontend (React)

- Single-page Application  
- Context API  
- WebSocket Client

## Backend (Express.js)

- Express Server  
- WebSocket Server

## Detection

-Mediapipe

- tenserflow (only planned)

# Key Components

## Core Logic

Each flashcard includes front, back, hint, and tags. Cards are scheduled using a spaced repetition algorithm. Then when trying to solve it the camera opens up and read the gestures.

## Frontend Components

- App.tsx: Main container  
- Card.tsx: Displays flashcard  
- ActionBar.tsx: Controls and WebSocket  
- Button.tsx: Reusable component  
- Header/Footer.tsx: Layout components

## Backend Server

- Express Server: Exposes /add-card  
- WebSocket Server: Sends updates

## Detection

* single file where it detects in live

## Gesture Recognition

Uses camera to detect:  
- 👍 Easy  
- ✋ Hard  
- 👎 Wrong

## Data Flow

1. Load cards  
2. Review  
3. Rate  
4. Update buckets  
5. Sync via WebSocket

## API Reference

POST /add-card  
Request:  
{  
 "front": "...",  
 "back": "...",  
 "hint": "...",  
 "tags": ["..."]  
}  
Response: { status: "ok" } or error

## WebSocket Events

- Connection: { message }  
- New Card: { type: 'new-card', card }

## CSS Styling

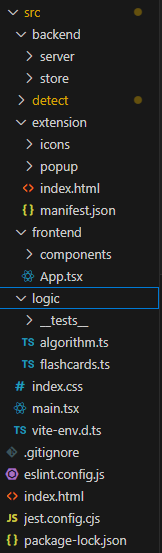
Uses CSS Modules per component:  
- Button.module.css  
- Card.module.css  
- Footer.module.css  
- Header.module.css  
- Main.module.css

## Getting Started

Prerequisites: Node.js, npm  
Install: git clone ...  
npm install  
Run backend: node server.js  
Run frontend: npm start

It also needs the mediapipe

## Development Notes



## Explanation of the code

It is written in the code. View comments.

## Extending the Application

- Add database ( after the amount of the flashcards gets bigger we will have to implement database instead of storing flashcards in ts)

- After showing the feedback it should go to the next flashcard  
- Import/export