**Curio: Fashion Assistant**

*MSc Computer Applications (Semester 1) — Solo Project*

**1. Title Page**

**Project Title:** Curio — Fashion Assistant  
**Submitted by:** Shraddha Tiwari  
**Course:** Master of Computer Applications (M.Sc. CA), Semester 1  
**Institution:** [Your College/University Name]  
**Academic Year:** 2025–26  
**Supervisor/Guide:** [Guide Name]

**2. Certificate**

This is to certify that the project report entitled **“Curio — Fashion Assistant”** submitted by **Shraddha Tiwari** in partial fulfillment of the requirements for the degree of **Master of Computer Applications (Semester 1)** is a record of the candidate’s original work carried out under my supervision and guidance.

**Signature of Guide**  
**Signature of Head of Department**

**3. Acknowledgement**

I would like to express my sincere gratitude to my project guide, **[Guide’s Name]**, for providing me with invaluable guidance, encouragement, and support throughout the development of this project.

I extend my thanks to the faculty members of the **Department of Computer Applications** for their assistance and for providing the necessary resources.

I am also grateful to my peers and family members for their continuous motivation and feedback, which helped me improve the quality of this work.

Finally, I acknowledge all the authors, researchers, and developers whose resources and tools I have referred to while working on **Curio**.

— *Shraddha Tiwari*

**Project Definition**

**Curio – Fashion Assistant** is a **MERN stack-based full-stack web application** designed to provide users with personalized fashion guidance and recommendations. The platform enables users to explore information about **face shapes, haircuts, hair accessories, body shapes, and outfit suggestions**, along with category-specific recommendations such as sandals, hand accessories, and neck accessories.

Visitors to the website can access the landing page to **view structured content**, including the Hero section, face-shape haircuts, hair accessories based on occasion, body-shape and build outfit recommendations, and category-based fashion suggestions. Registered users gain additional interactive functionalities, allowing them to **calculate their face shape and body metrics, discover personalized hairstyles and outfit combinations, save favorite items**, and manage their profile preferences.

The system includes an **admin module** for content management, enabling admins to add, edit, or delete recommendations, manage user accounts, and ensure that the platform remains up-to-date and relevant.

The project emphasizes a **user-centric design**, interactivity, and scalability, combining modern web technologies to deliver an **intuitive and responsive fashion assistant**. Curio not only provides informational browsing but also transforms it into a personalized experience, helping users make informed fashion choices while offering scope for future enhancements such as AI-driven recommendations, virtual try-on features, and mobile accessibility.

**Abstract**

Curio – Fashion Assistant is a **full-stack web application** developed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, designed to provide users with an interactive and personalized fashion discovery experience. The platform aims to bridge the gap between traditional informational websites and personalized fashion guidance, offering both general information and tailored recommendations.

Visitors to the website can access the **landing page**, which includes a structured layout with a **header, footer, and main content** divided into four key sections: the **Hero Section** for introductory content, **Face Shape & Haircuts** providing information on different face types and suitable haircuts, **Hair Accessories** suggesting appropriate accessories for various occasions, and **Body Shape, Height & Build** displaying general outfit recommendations. In addition, category-based suggestions, such as sandals, hand accessories, and neck accessories, are presented for general browsing. Visitors can also navigate to the **login and registration pages** to create an account for a personalized experience.

Registered users gain enhanced functionality, allowing them to **calculate their face shape and body metrics**, discover suitable hairstyles, accessories, and outfit recommendations based on their personal attributes, and save their preferences for future reference. The system provides an intuitive search and filter mechanism for efficient navigation across different categories.

The **admin module** ensures seamless management of website content, including haircuts, accessories, outfit recommendations, and category-specific items, as well as user accounts and activity monitoring.

Curio demonstrates the effective use of modern web technologies to combine information, interactivity, and personalization. The project emphasizes a **user-centric design, scalability, and responsiveness**, making fashion discovery easier, engaging, and tailored to individual preferences. The platform has the potential to expand into more advanced features such as AI-driven recommendations, virtual try-ons, and mobile applications.

**Introduction**

Fashion is one of the most dynamic and rapidly evolving industries, with trends changing frequently and consumer preferences becoming increasingly personalized. With the advent of digital platforms, the way people discover, evaluate, and purchase fashion products has shifted significantly from physical stores to online mediums. While e-commerce platforms provide extensive catalogs of products, they often lack personalization and interactivity, leaving users overwhelmed and unsure about what suits their style, body type, or occasion.

**Curio – Fashion Assistant** addresses these challenges by providing a **web-based platform** that combines fashion information with personalized discovery. Developed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, Curio offers both **informational and interactive features**. Visitors to the website can explore structured content including face shapes and suitable haircuts, hair accessories for different occasions, body shapes with corresponding outfit recommendations, and category-based suggestions such as sandals, hand, or neck accessories. The website also provides login and registration functionality to enable personalized experiences.

Registered users gain the ability to **interactively calculate their face shape and body metrics**, discover personalized haircuts, accessories, and outfits, and save favorites for future reference. This personalization enhances engagement and helps users make informed fashion choices based on their unique attributes.

From a technical perspective, Curio leverages **React.js** for a dynamic and responsive frontend, **Node.js and Express.js** for a robust backend, and **MongoDB** for flexible and scalable data management. The system also includes an **admin module** for content management and monitoring, ensuring the platform remains accurate, relevant, and user-friendly.

Overall, Curio demonstrates the effective integration of modern web technologies to create a **user-centric, scalable, and interactive fashion assistant**, bridging the gap between general information and personalized recommendations in the fashion domain.

## ****Objectives****

The primary objective of the **Curio – Fashion Assistant** project is to design and implement a **full-stack web application** that offers personalized fashion discovery while maintaining a user-friendly interface. The system aims to bridge the gap between general fashion information and interactive, user-centric recommendations.

### ****Specific Objectives****

1. **Design a responsive and intuitive user interface**
   * Develop the frontend using **React.js** to ensure smooth navigation across different sections, including face shapes, haircuts, hair accessories, body shapes, and outfit recommendations.
   * Provide a visually appealing layout with structured content for easy browsing.
2. **Implement personalized fashion discovery for users**
   * Enable registered users to **calculate their face shape and body metrics**.
   * Provide **interactive recommendations** for haircuts, accessories, and outfits based on individual attributes.
   * Allow users to **save favorites** for future reference.
3. **Build a robust backend**
   * Use **Node.js and Express.js** to create RESTful APIs for data management and user interactions.
   * Ensure secure authentication and session management for registered users.
4. **Integrate a scalable database**
   * Implement **MongoDB** to manage users, products, recommendations, ratings, and favorites.
   * Ensure flexibility for handling dynamic data such as personalized recommendations and category-based suggestions.
5. **Provide administrative control**
   * Develop an **admin module** to manage website content, user accounts, and recommendations efficiently.
   * Allow admins to add, edit, or delete information for haircuts, accessories, outfits, and category-specific items.
6. **Enhance usability and engagement**
   * Implement search and filter functionalities for easier navigation.
   * Ensure a responsive and accessible design across devices.

## ****System Analysis****

### ****1. Problem Definition****

The fashion industry is rapidly evolving, with consumers seeking personalized guidance for style choices. Existing platforms often provide large catalogs of fashion products but lack **interactive and personalized features**, leaving users overwhelmed and uncertain about what suits their face shape, body type, or occasion. Users struggle to:

* Determine the best **haircuts** for their face shape.
* Select **appropriate hair accessories** for specific occasions.
* Choose outfits that match their **body shape, height, and build**.
* Receive **category-based fashion recommendations** in a consolidated, user-friendly format.

There is a need for a **web application** that combines informational content with interactive tools, enabling users to **calculate, discover, and save personalized fashion recommendations** efficiently.

### ****2. Existing System****

Current e-commerce or fashion information websites generally provide:

* Large catalogs of fashion items.
* Basic search and filter options.
* Static advice about trends and styles.

**Limitations:**

* Lack of interactive tools for personal measurements.
* No personalized recommendations based on face shape or body type.
* Users cannot save favorites or track recommendations.
* Admins often cannot manage content dynamically, leading to outdated information.

### ****3. Proposed System (Curio – Fashion Assistant)****

The proposed system provides a **personalized and interactive fashion assistant** with:

* **Landing page** with structured sections: Hero, Face Shape & Haircuts, Hair Accessories, Body Shape & Outfit Recommendations, and category-based suggestions.
* **Interactive tools** for registered users to calculate face shape, body metrics, and discover personalized recommendations.
* **Save favorites** functionality for haircuts, outfits, and accessories.
* **Admin module** to manage content and user accounts dynamically.

### ****4. Feasibility Study****

**a) Technical Feasibility**

* MERN stack ensures a **fully integrated, scalable, and responsive application**.
* React.js provides dynamic UI, while Node.js and Express.js offer robust backend services.
* MongoDB enables flexible data storage for user preferences, recommendations, and content.

**b) Economic Feasibility**

* Open-source technologies reduce costs.
* Minimal hardware requirements; project is feasible within student resources.

**c) Operational Feasibility**

* Intuitive interface ensures easy navigation for both visitors and registered users.
* Admin module allows straightforward content management.

**d) Time Feasibility**

* Modular development (frontend, backend, database) allows completion within the semester timeline.
* Phased implementation ensures smooth progress without bottlenecks.

## ****System Design****

The system design of **Curio – Fashion Assistant** focuses on creating a scalable, modular, and user-friendly application. It outlines the architecture, data flow, and interaction between different components, ensuring smooth functionality for visitors, registered users, and admins.

### ****1. System Architecture****

The project follows a **3-tier architecture**:

1. **Presentation Layer (Frontend)**
   * Built with **React.js**, this layer provides a **dynamic and responsive user interface**.
   * Components include: Header, Footer, Hero Section, Face Shape & Haircuts, Hair Accessories, Body Shape & Outfit Recommendations, and Category-Based Recommendations.
   * Handles user input for interactive features like face shape calculation, body metrics input, and saving favorites.
2. **Application Layer (Backend)**
   * Developed using **Node.js** and **Express.js**, this layer handles all **business logic**, API requests, and server-side operations.
   * Responsibilities include:
     + User authentication and session management.
     + CRUD operations for recommendations, haircuts, outfits, and accessories.
     + Processing calculations for personalized recommendations.
3. **Data Layer (Database)**
   * Implemented using **MongoDB** for **flexible and scalable data storage**.
   * Collections include: Users, Recommendations, Ratings, Favorites, Face Shapes, Body Metrics, Outfits, and Admin data.
   * Mongoose ODM manages schema definitions and database interactions.

### ****2. Data Flow****

* **Visitors** request pages; the frontend fetches static information from the backend.
* **Registered Users** interact with input forms; frontend sends data to backend APIs, which process calculations and fetch personalized recommendations from MongoDB.
* **Admin** performs content management operations; backend updates the database accordingly.
* Responses are sent back to the frontend for rendering dynamic content.

### ****3. Use Case Description****

**Actors:** Visitor, Registered User, Admin

| **Actor** | **Use Case** | **Description** |
| --- | --- | --- |
| Visitor | View Content | Access landing page and explore informational sections. |
| Registered User | Calculate & Discover | Input face/body metrics to get personalized haircuts, accessories, and outfits. |
| Registered User | Save Favorites | Save selected recommendations for future reference. |
| Admin | Manage Content | Add/Edit/Delete haircuts, accessories, outfits, and category-based items. |
| Admin | Manage Users | View, block, or delete user accounts as needed. |

### ****4. Entity-Relationship (ER) Overview****

* **Users** ↔ **Favorites** (one-to-many)
* **Users** ↔ **Outfits** (one-to-many)
* **Products/Recommendations** ↔ **Ratings** (one-to-many)
* Admin manages all collections (Users, Recommendations, Accessories, Outfits)

(ER diagram to be included in the report as a figure, showing collections and relationships.)

## ****Methodology****

The **methodology** outlines the structured approach followed during the development of the **Curio – Fashion Assistant** web application. The project was developed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** and followed a **systematic software development lifecycle** to ensure a robust, scalable, and user-friendly solution.

### ****1. Requirement Gathering****

* **Functional Requirements:**
  + Landing page with header, footer, and main content sections (Hero, Face Shape & Haircuts, Hair Accessories, Body Shape & Outfit Recommendations, Category-based Recommendations).
  + Visitor functionalities: view information, access login/registration.
  + Registered user functionalities: calculate face/body metrics, discover personalized recommendations, save favorites.
  + Admin functionalities: manage content, users, and recommendations.
* **Non-Functional Requirements:**
  + Responsive and intuitive UI.
  + Secure user authentication.
  + Scalable and maintainable architecture.
* **Tools & Technologies:**
  + Frontend: React.js, CSS, HTML
  + Backend: Node.js, Express.js
  + Database: MongoDB
  + Version Control: Git & GitHub

### ****2. System Design****

* Defined a **3-tier architecture** (Frontend, Backend, Database).
* Created **wireframes and layouts** for all main pages.
* Designed **ER diagram** for database collections: Users, Favorites, Recommendations, Ratings, Outfits.
* Defined API endpoints for communication between frontend and backend.

### ****3. Development / Implementation****

* **Frontend Development:**
  + Built reusable components for Hero Section, Product/Recommendation Cards, Rating System, and Favorites.
  + Implemented interactive calculations for face and body metrics.
* **Backend Development:**
  + Created RESTful APIs for user authentication, CRUD operations on recommendations, favorites, and ratings.
  + Implemented business logic for personalized recommendations.
* **Database Integration:**
  + Designed MongoDB schemas and relationships using Mongoose.
  + Ensured data consistency and optimized queries for faster response.

### ****4. Testing****

* Conducted **unit testing** for individual components and APIs.
* Performed **integration testing** to ensure smooth frontend-backend communication.
* Verified **user flows**: Visitor browsing, user calculations, admin content management.

### ****5. Deployment (Future Scope)****

* Frontend deployment: Netlify or Vercel.
* Backend deployment: Heroku or Render.
* Database: MongoDB Atlas (cloud-based).

**References**

1. Shama, U., & Agrawal, R. (2021). *Full-Stack Web Development with MERN*. Apress.
2. Grinberg, M. (2018). *Flask Web Development*. O’Reilly Media. *(for REST API concepts)*
3. Kumar, A. (2020). *Learning React: Modern Patterns for Developing React Apps*. O’Reilly Media.
4. Holmes, A. (2019). *Node.js 8 the Right Way: Practical, Server-Side JavaScript That Scales*. Pragmatic Bookshelf.
5. Chodorow, K. (2013). *MongoDB: The Definitive Guide*. O’Reilly Media.
6. Pressman, R. S. (2014). *Software Engineering: A Practitioner’s Approach*. McGraw Hill.
7. Sommerville, I. (2015). *Software Engineering* (10th Edition). Pearson.
8. Myntra. (2024). *Fashion E-commerce Platform*. [https://www.myntra.com](https://www.myntra.com/)
9. MongoDB Documentation. (2025). *MongoDB Manual*. <https://www.mongodb.com/docs>
10. ReactJS Documentation. (2025). *React Official Docs*. [https://react.dev](https://react.dev/)
11. Node.js Documentation. (2025). *Node.js Docs*. [https://nodejs.org](https://nodejs.org/)
12. Express.js Documentation. (2025). *Express Guide*. [https://expressjs.com](https://expressjs.com/)

**1. Visitor Functionalities (Guest / Unregistered)**

A visitor can:

1. Access the **landing page** with header, footer, and main content.
2. View **Hero Section** – introductory content/banner.
3. View **Face Shape & Haircuts** – basic information about different face shapes and suggested haircuts (read-only).
4. View **Hair Accessories** – information on accessories for different occasions.
5. View **Body Shape, Height & Build** – general outfit recommendations based on body metrics.
6. View **Category-based Recommendations** – e.g., sandals, hand accessories, neck accessories.
7. Access **login/registration page** to create credentials for personalized interaction.

*Note:* Visitors **cannot calculate or interact** with these recommendations — they can only view the information.

**2. Registered User Functionalities (Logged-in)**

A logged-in user can:

1. Access all **Visitor functionalities** with added personalization.
2. Use the **interactive discovery features**:
   * **Calculate face shape** (if inputting measurements or using guided selection).
   * **Discover suitable haircuts** based on personal face shape.
   * **Choose hair accessories** based on occasions they select.
   * **Get personalized outfit recommendations** based on their body shape, height, and build.
3. Save preferences and **favorites** for haircuts, accessories, and outfits.
4. Search and filter **recommendations** by category (sandals, hand/neck accessories, etc.) or by occasion.
5. View a **profile/dashboard** with saved items and past discoveries.
6. Log out securely.

*Note:* The **key difference from visitors** is that users can **interactively calculate, discover, and save personalized recommendations**.

**3. Admin Functionalities**

An admin can:

1. Log in through a **secure admin portal**.
2. **Manage main content sections**:
   * Hero section, Face Shape & Haircuts, Hair Accessories, Body Shape & Outfit Recommendations, Category-based Recommendations.
3. **Add, edit, or delete**:
   * Face-shape haircuts
   * Hair accessories per occasion
   * Outfit recommendations based on body type
   * Category-based items (sandals, hand accessories, neck accessories, etc.)
4. **Manage user accounts** (view, block, or delete).
5. Monitor ratings, favorites, and user interactions (optional).
6. Ensure content is **accurate and up-to-date**.

**Functionality Table: Visitor, User, Admin**

| **Role** | **Functionality / Action** | **Details / Notes** |
| --- | --- | --- |
| **Visitor** | Access landing page | View header, footer, and main content sections |
|  | View Hero section | Introductory banner or content |
|  | View Face Shape & Haircuts | Read-only info about face shapes and suggested haircuts |
|  | View Hair Accessories | Information about hair accessories based on occasions |
|  | View Body Shape, Height & Build | General outfit recommendations |
|  | View Category-based Recommendations | Sandals, hand accessories, neck accessories, etc. |
|  | Access Login / Registration | Create credentials for interactive features |
| **User** | All Visitor functionalities | Plus interactive features |
|  | Calculate Face Shape | Enter details or select options to identify personal face shape |
|  | Discover Suitable Haircuts | Personalized haircut recommendations based on face shape |
|  | Select Hair Accessories | Personalized accessory suggestions based on occasion |
|  | Personalized Outfit Recommendations | Based on body shape, height, and build |
|  | Save Favorites | Save haircuts, accessories, outfits for future reference |
|  | Search & Filter Recommendations | Filter by category or occasion |
|  | Profile / Dashboard | View saved items, past discoveries |
|  | Logout | Secure exit |
| **Admin** | Secure Admin Login | Access admin portal |
|  | Manage Hero Section | Add/edit/delete hero/banner content |
|  | Manage Face Shape & Haircuts Section | Add/edit/delete face-shape based haircut info |
|  | Manage Hair Accessories Section | Add/edit/delete accessories per occasion |
|  | Manage Body Shape & Outfit Recommendations | Add/edit/delete outfit suggestions per body type |
|  | Manage Category-based Recommendations | Add/edit/delete sandals, hand/neck accessories |
|  | Manage Users | View, block, or delete user accounts |
|  | Monitor Ratings / Favorites (optional) | Ensure system content quality |
|  | Maintain Content Accuracy | Keep recommendations and information updated |

**1. Visitor Functionalities (Guest / Unregistered)**

1. Access the **landing page** with header, footer, and main content.
2. View **Hero Section** – introductory banner/content.
3. View **Face Shape & Haircuts** – basic information about different face types and suggested haircuts (read-only).
4. View **Hair Accessories** – recommendations for various occasions (read-only).
5. View **Body Shape, Height & Build** – general outfit recommendations.
6. View **Category-based Recommendations** – e.g., sandals, hand accessories, neck accessories.
7. Access **Login/Registration** page to create an account for personalized features.

*Note:* Visitors **cannot calculate, interact, or save recommendations**.

**2. Registered User Functionalities**

1. Access all **Visitor functionalities**.
2. **User Authentication:** Register, login, and logout securely.
3. **Calculate Face Shape** – input or select options to identify personal face type.
4. **Discover Haircuts** – personalized recommendations based on face shape.
5. **Select Hair Accessories** – personalized suggestions based on occasions.
6. **Get Outfit Recommendations** – tailored to body shape, height, and build.
7. **Save Favorites** – haircuts, accessories, outfits for future reference.
8. **Search and Filter** – explore recommendations by category or occasion.
9. View **Profile/Dashboard** – previously saved favorites and personalized discoveries.

**3. Admin Functionalities**

1. **Secure Admin Login** – access to the admin portal.
2. **Manage Content Sections:**
   * Hero section
   * Face Shape & Haircuts
   * Hair Accessories
   * Body Shape & Outfit Recommendations
   * Category-based Recommendations (sandals, hand/neck accessories, etc.)
3. **CRUD Operations:** Add, edit, or delete content in all sections.
4. **Manage Users:** View, block, or delete user accounts as necessary.
5. **Monitor Ratings / Favorites** – ensure content relevance and quality.
6. Ensure **website content is accurate and up-to-date**.
7. Perfect! Let’s first create a **high-level directory structure** for your **Curio MERN stack project**, showing main folders and files. Then, we can break down **each folder in detail**.

## ****1. High-Level Directory Structure (Main Folders & Files)****

Curio/

│

├── client/ # Frontend (React.js)

│ ├── public/

│ └── src/

│ ├── assets/

│ ├── components/

│ ├── pages/

│ ├── sections/

│ ├── services/

│ ├── App.jsx

│ └── index.js

│

├── server/ # Backend (Node.js + Express.js)

│ ├── controllers/

│ ├── models/

│ ├── routes/

│ ├── middleware/

│ ├── utils/

│ ├── config/

│ └── server.js

│

├── database/ # Database setup / seed scripts

│ └── seed.js

│

├── .gitignore

├── package.json

└── README.md

## ****2. Detailed Directory Structure for Each Folder****

### ****A. client/**** (Frontend)

client/

├── public/

│ ├── index.html

│ └── favicon.ico

│

├── src/

│ ├── assets/ # Images, icons, fonts, static resources

│ ├── components/ # Reusable components like Header, Footer, Card, Rating

│ ├── pages/ # Full pages like Home.jsx, Login.jsx, Registration.jsx, Dashboard.jsx

│ ├── sections/ # Sections used within pages, e.g., Hero.jsx, FaceShapeSection.jsx

│ ├── services/ # API service calls, e.g., userService.js, productService.js

│ ├── App.jsx # Main React component

│ └── index.js # Entry point rendering App.jsx

### ****B. server/**** (Backend)

server/

├── controllers/ # Logic for handling requests

│ ├── userController.js

│ ├── adminController.js

│ ├── recommendationController.js

│ └── authController.js

│

├── models/ # MongoDB schemas

│ ├── User.js

│ ├── Admin.js

│ ├── Recommendation.js

│ ├── Rating.js

│ ├── Favorite.js

│ └── Outfit.js

│

├── routes/ # API routes

│ ├── userRoutes.js

│ ├── adminRoutes.js

│ └── recommendationRoutes.js

│

├── middleware/ # Middleware like auth, error handling

│ ├── authMiddleware.js

│ └── errorMiddleware.js

│

├── utils/ # Helper functions

│ ├── calculateFaceShape.js

│ ├── bodyRecommendation.js

│ └── sendEmail.js

│

├── config/ # Configuration files

│ └── db.js # MongoDB connection setup

│

└── server.js # Entry point for backend server

### ****C. database/****

database/

└── seed.js # Seed initial data like default face shapes, accessories, outfits

### ****D. Root Files****

.gitignore # Ignore node\_modules, env files, etc.

package.json # Project dependencies

README.md # Project overview

**Directory Structure**

**1. client/ (Frontend)**

client/

├── public/

│ ├── index.html

│ ├── favicon.ico

│ └── manifest.json

│

├── src/

│ ├── assets/

│ │ ├── images/ # Hero images, face shapes, body/outfit images

│ │ └── icons/ # Accessory icons, UI icons

│ │

│ ├── components/ # Reusable UI components

│ │ ├── Header.jsx

│ │ ├── Footer.jsx

│ │ ├── Card.jsx # Body type / outfit / accessory card

│ │ ├── Rating.jsx # Star rating component

│ │ ├── SearchBar.jsx

│ │ └── Loader.jsx

│ │

│ ├── sections/ # Sections used in pages

│ │ ├── HeroSection.jsx

│ │ ├── FaceShapeSection.jsx

│ │ ├── HaircutSection.jsx

│ │ ├── HairAccessorySection.jsx

│ │ ├── BodyShapeSection.jsx

│ │ ├── OutfitRecommendationSection.jsx

│ │ └── CategoryRecommendationSection.jsx

│ │

│ ├── pages/ # Full pages

│ │ ├── Home.jsx

│ │ ├── Login.jsx

│ │ ├── Registration.jsx

│ │ ├── Dashboard.jsx # User-specific saved favorites & recommendations

│ │ ├── AdminDashboard.jsx

│ │ └── NotFound.jsx

│ │

│ ├── services/ # API service calls

│ │ ├── userService.js

│ │ ├── authService.js

│ │ ├── recommendationService.js

│ │ └── adminService.js

│ │

│ ├── context/ # Optional React context for global state

│ │ └── AuthContext.jsx

│ │

│ ├── App.jsx # Main App component with routing

│ └── index.js # React entry point

**2. server/ (Backend)**

server/

├── controllers/ # Request handlers

│ ├── userController.js # Handles user registration, login, dashboard

│ ├── adminController.js # Admin CRUD operations

│ ├── recommendationController.js # Handles fetching recommendations

│ ├── faceShapeController.js # Face shape calculations

│ └── outfitController.js # Outfit generation logic

│

├── models/ # MongoDB schemas

│ ├── User.js # User info, credentials, saved favorites

│ ├── Admin.js

│ ├── Recommendation.js # Haircuts, accessories, outfits

│ ├── Rating.js

│ ├── Favorite.js

│ └── Outfit.js

│

├── routes/ # API endpoints

│ ├── userRoutes.js # /api/users

│ ├── adminRoutes.js # /api/admin

│ ├── recommendationRoutes.js # /api/recommendations

│ └── authRoutes.js # /api/auth

│

├── middleware/ # Custom middleware

│ ├── authMiddleware.js # JWT auth validation

│ ├── adminMiddleware.js

│ └── errorMiddleware.js

│

├── utils/ # Helper functions

│ ├── calculateFaceShape.js

│ ├── bodyRecommendation.js

│ └── sendEmail.js

│

├── config/ # Configuration files

│ └── db.js # MongoDB connection

│

└── server.js # Express server entry point

**3. database/**

database/

└── seed.js # Script to populate initial face shapes, haircuts, outfits, accessories

**4. Root Files**

.gitignore # Ignore node\_modules, .env, etc.

package.json # Project dependencies (for both server & client if using monorepo)

README.md # Project overview & instructions

.env # Environment variables (API keys, DB URIs)

## ****Curio Database Schema (ER Diagram Description)****

### ****Collections and Relationships****

1. **Users**

* \_id → Unique identifier
* name, email, password, faceShape, bodyShape, height, build
* favorites → references **Favorites** collection
* **Relationships:**
  + One-to-many → Favorites
  + One-to-many → Ratings

1. **Admins**

* \_id, name, email, password, role
* Manages content in: **FaceShapes, BodyShapes, Recommendations, Outfits**

1. **FaceShapes**

* \_id, name, description, imageUrl
* recommendedHaircuts → references **Recommendations** (type = haircut)
* **Relationships:**
  + One-to-many → Recommendations (Haircuts)

1. **BodyShapes**

* \_id, name, heightRange, build, description
* recommendedOutfits → references **Outfits** collection
* **Relationships:**
  + One-to-many → Outfits

1. **Recommendations**

* \_id, type (haircut, accessory, outfit), category, name, description, applicableFaceShape, applicableBodyShape, occasions, imageUrl
* **Relationships:**
  + Many-to-one → FaceShapes (haircuts)
  + Many-to-one → BodyShapes (outfits)

1. **Outfits**

* \_id, name, description, bodyShape, heightRange, build, category, imageUrl
* **Relationships:**
  + Many-to-one → BodyShapes

1. **Favorites**

* \_id, userId → Users, recommendationId → Recommendations, createdAt

1. **Ratings**

* \_id, userId → Users, recommendationId → Recommendations, rating, comment, createdAt

### ****Relationship Summary****

| **From Collection** | **To Collection** | **Relationship Type** | **Purpose** |
| --- | --- | --- | --- |
| Users | Favorites | One-to-Many | Store items saved by users |
| Users | Ratings | One-to-Many | Store user ratings/comments |
| FaceShapes | Recommendations | One-to-Many | Haircuts linked to face shapes |
| BodyShapes | Outfits | One-to-Many | Outfits linked to body shapes |
| Recommendations | FaceShapes/BodyShapes | Many-to-One | Reverse link for queries |
| Favorites | Recommendations | Many-to-One | Identify saved recommendations |
| Ratings | Recommendations | Many-to-One | Store user ratings per recommendation |

### ****Diagram Representation**** (text version for your doc)

Users --< Favorites >-- Recommendations

Users --< Ratings >-- Recommendations

FaceShapes --< Recommendations (Haircuts)

BodyShapes --< Outfits

Recommendations -- faceShape/bodyShape

Outfits -- bodyShape

Admins -- manage all collections

**1. Users Table**

**Purpose:** Stores registered user information and saved preferences.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| name | String | User full name |
| email | String | User email (unique) |
| password | String | Hashed password |
| faceShape | String | Calculated or selected face shape |
| bodyShape | String | Calculated or selected body shape |
| height | Number | User height |
| build | String | Body build description (Slim, Athletic, etc.) |
| favorites | [ObjectId] | Array referencing Favorites collection |
| createdAt | Date | Account creation timestamp |
| updatedAt | Date | Last update timestamp |

**2. Admins Table**

**Purpose:** Stores admin account details for content management.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| name | String | Admin full name |
| email | String | Admin email |
| password | String | Hashed password |
| role | String | Default: "admin" |
| createdAt | Date | Account creation timestamp |
| updatedAt | Date | Last update timestamp |

**3. FaceShapes Table**

**Purpose:** Stores face shapes and associated haircut recommendations.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| name | String | Face shape name (Oval, Round, Heart, etc.) |
| description | String | Detailed info about face shape |
| recommendedHaircuts | [ObjectId] | References Recommendations collection (type = haircut) |
| imageUrl | String | Image representing the face shape |
| createdAt | Date | Timestamp |
| updatedAt | Date | Timestamp |

**4. BodyShapes Table**

**Purpose:** Stores body types, height/build info, and outfit recommendations.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| name | String | Body shape name (Pear, Apple, Hourglass, etc.) |
| heightRange | String | Height range (e.g., 150–170 cm) |
| build | String | Build type (Slim, Athletic, etc.) |
| description | String | Body shape details |
| recommendedOutfits | [ObjectId] | References Outfits collection |
| createdAt | Date | Timestamp |
| updatedAt | Date | Timestamp |

**5. Recommendations Table**

**Purpose:** Stores haircuts, accessories, and category-based recommendations.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| type | String | Haircut / Accessory / Outfit |
| category | String | E.g., sandals, hand accessory, neck accessory |
| name | String | Name of recommendation |
| description | String | Short description |
| applicableFaceShape | [String] | Face shapes applicable |
| applicableBodyShape | [String] | Body shapes applicable |
| occasions | [String] | Recommended occasions |
| imageUrl | String | Image representing the item |
| createdAt | Date | Timestamp |
| updatedAt | Date | Timestamp |

**6. Outfits Table**

**Purpose:** Stores outfits tailored to body types, height, and build.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| name | String | Outfit name |
| description | String | Outfit details |
| bodyShape | [String] | Targeted body shapes |
| heightRange | String | Applicable height range |
| build | String | Applicable build type |
| category | String | Outfit category (Casual, Formal, etc.) |
| imageUrl | String | Image of outfit |
| createdAt | Date | Timestamp |
| updatedAt | Date | Timestamp |

**7. Favorites Table**

**Purpose:** Stores items saved by users.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| userId | ObjectId | Reference to Users collection |
| recommendationId | ObjectId | Reference to Recommendations collection |
| createdAt | Date | Timestamp |

**8. Ratings Table**

**Purpose:** Stores user ratings and feedback for recommendations.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| \_id | ObjectId | Unique identifier |
| userId | ObjectId | Reference to Users collection |
| recommendationId | ObjectId | Reference to Recommendations collection |
| rating | Number | Rating value (1–5) |
| comment | String | Optional feedback |
| createdAt | Date | Timestamp |

**Curio – Fashion Assistant: Project Concept**

**Project Concept:**  
Curio is an interactive web application designed as a **personalized fashion assistant** that leverages the **MERN (MongoDB, Express.js, React.js, Node.js) stack** to deliver customized hairstyle, accessory, and outfit recommendations based on individual face shapes, body types, height, and build. The system aims to empower users to make informed fashion choices while providing a seamless, dynamic, and engaging user experience.

**Purpose and Objective:**  
The primary objective of Curio is to bridge the gap between generic fashion suggestions and personalized style guidance. Users can explore hairstyle suggestions, hair accessories, outfit recommendations, and category-based fashion items (e.g., sandals, hand accessories, neck accessories) tailored specifically to their physical attributes. Visitors can browse informational content without registration, while registered users gain access to interactive tools that allow them to calculate their face shape, body type, and discover fashion recommendations suited to their individual features. Admins manage all content dynamically, ensuring updated and relevant fashion suggestions.

**User Roles and Functionality:**

1. **Visitor:** Can access the landing page with structured content sections including Hero, Face Shape, Haircuts, Hair Accessories, Body & Outfit Recommendations, and Category-based recommendations. Visitors can read informational content but cannot personalize suggestions.
2. **Registered User:** Can calculate face shape and body type metrics, receive personalized recommendations, save favorite items, rate and review suggestions, and filter/search recommendations by categories or occasions.
3. **Admin:** Can perform CRUD operations on all content sections including Face Shapes, Body Shapes, Haircuts, Accessories, Outfits, and Category-based Recommendations, as well as manage user accounts to ensure a dynamic and updated system.

**Technical Overview:**

* **Frontend:** Built with React.js, featuring modular components and sections for each functional area. Components like Header, Footer, Cards, Rating, and SearchBar ensure a responsive, interactive, and intuitive UI.
* **Backend:** Node.js with Express.js handles API requests, authentication, recommendation logic, and dynamic content management.
* **Database:** MongoDB stores all dynamic data in collections including Users, Admins, FaceShapes, BodyShapes, Recommendations, Outfits, Favorites, and Ratings, with relationships designed to support personalization and interactivity.
* **Features:** Dynamic calculations, personalized recommendations, save favorites, rating system, search and filter, and admin-controlled content updates.

**Impact and Significance:**  
Curio transforms the fashion exploration experience by combining personalization with a user-friendly interface. It demonstrates the practical application of modern web development technologies to solve real-world problems in the fashion domain, offering scalability for future enhancements like AI-driven style recommendations, virtual try-on, or mobile app integration.