[Trendly]

A Project Report for Industrial Training and Internship

submitted by

[Lushi Gre]

In the partial fulfillment of the award of the degree of

B.Tech

in the

[CSE]

Of

[Sandip university]

At

Ardent Computech Pvt. Ltd.

CERTIFICATE FROM SUPERVISOR

This is to certify that **“Lushi Gre, 220205131097**” have completed the project titled "Trendly" under my supervision during the period from “<3/07/2025>” to “2/08/2025” which is in partial fulfillment of requirements for the award of the **B.Tech** degree and submitted to the Department of “**CSE**” of **“Sandip University”**.

_____

Signature of the Supervisor

Date: 10/08/2025

Name of the Project Supervisor: Subhojit Santra

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

"Trendly" is the bonafide work of

Name of the student: Lushi Gre ***Signature:***

SIGNATURE

Name :

PROJECT MENTOR

Subhojit santra

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***Subhojit santra*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

CONTENT PAGE

1. INTRODUCTION

In today's fast-paced digital marketplace, online shopping has become the preferred method of purchasing goods and services. Millions of customers browse e-commerce platforms daily in search of the best deals. However, one of the biggest challenges faced by shoppers is the constant fluctuation of product prices. Prices may drop unexpectedly for a limited time, and customers often miss these opportunities, resulting in overspending or delayed purchases.

To address this gap, **Trendly** has been developed as a comprehensive web application powered by the **MERN stack (MongoDB, Express.js, React.js, Node.js)**. The platform is designed to help online shoppers track product prices, analyze historical price trends, and receive instant alerts whenever their desired product reaches a target price or experiences a price drop. By combining modern web technologies with real-time data monitoring, Trendly empowers users to make smarter and more cost-effective purchasing decisions.

The system provides a **user-friendly and interactive interface** where customers can search for products, add them to a personal watchlist, and visualize their price history through intuitive charts. A built-in scheduler continuously monitors product prices and automatically notifies users via email alerts when the conditions match their preferences. Furthermore, Trendly features an analytics dashboard that highlights trending products, most watchlisted items, and price drop frequencies, thereby giving users valuable market insights.

This project not only enhances transparency in online shopping but also ensures that users stay informed and never miss out on the best deals. By automating the price-tracking process, Trendly eliminates the manual effort of constantly checking multiple websites for updates. It ultimately bridges the gap between consumers and real-time market dynamics, making online shopping more **efficient, data-driven, and user-centric**.

2. SYSTEM ANALYSIS

2A. OBJECTIVE

The primary objective of **Trendly** is to transform the way consumers make purchase decisions online by delivering a transparent, accessible, and data-driven price-tracking platform. Trendly minimizes overspending and FOMO (fear of missing out) by surfacing reliable price history, real-time price changes, and proactive alerts tailored to each user.

More specifically, Trendly aims to:

- **Provide price transparency** through historical charts and clear metadata (current price, last updated, source/URL).
- **Deliver timely notifications** via email (and extensible to push/WhatsApp) when prices drop or reach a user's **target price**.
- **Personalize shopping decisions** with a configurable watchlist, category preferences, and alert thresholds.
- **Offer market insights** (trending products, most-watchlisted items, price-drop frequency) to guide smarter purchases.
- **Ensure security and reliability** using JWT authentication, bcrypt password hashing, scheduled checks with monitoring, and robust error handling.

Ultimately, Trendly empowers users to **save money and time—anytime, anywhere**, by bridging the gap between fluctuating online prices and actionable, real-time information.

2B. SCOPE

Our project focuses on a **web-based application** (responsive across mobile and desktop) that tracks prices across supported e-commerce sites, stores price history, and notifies users on meaningful changes. The scope for the current release includes:

1. **User Registration & Login**
    - Secure sign-up/login with **JWT**; passwords hashed using **bcrypt**.
    - Profile management: name, email, timezone, preferred currency, notification preferences.
    - Session management, rate-limiting, and basic account recovery.
2. **Product Search & Discovery**
    - Global search bar with filters (category, price range).
    - Product cards with **image, title, current price, source, last updated**.
    - **Trending** and **Most-Watchlisted** sections for discovery.
3. **Watchlist System**
    - Add/remove any product to a personal watchlist.
    - Toggle **Enable Price Drop Alert** and set **Target Price** (e.g., "Notify me when this drops below ₹1000").
    - De-duplication and basic validation of product URLs; per-product alert frequency controls.
4. **Admin Panel**
    - Manage products, categories, and data quality (edit/verify product metadata).
    - Configure **scraper/scheduler** frequency and sources; view logs and job status.
    - User management (basic roles: admin/moderator) and system health metrics.
5. **Multiplatform Access**
    - **Responsive UI** with mobile-first design (TailwindCSS) and smooth interactions (Framer Motion).
    - **PWA** support (installable on devices) and offline caching for recent price charts (where feasible).
    - **Extensible API** for future Android/iOS apps and a **browser extension** (one-click "Add to Trendly").
6. **Payment Integration** *(optional / for premium features)*
    - Freemium model with optional subscriptions for advanced alerts (SMS/WhatsApp), higher check frequency, and unlimited watchlists.
    - Integration readiness for payment gateways (e.g., Razorpay/Stripe) with invoices and plan management.

Out of Scope (current phase):

- Direct product purchasing or checkout within Trendly.
- Price prediction using AI/ML (planned as a future enhancement).

Price tracking for unsupported sites or those that disallow automated access (we adhere to applicable TOS and legal constraints**).**

3. SYSTEM ANALYSIS

3A. IDENTIFICATION OF NEED

System analysis is one of the most critical phases in the development of any project, and for **Trendly** it plays an especially important role because the system is designed to meet the needs of a diverse group of online shoppers. The ultimate goal of this stage is to carefully evaluate the problem, identify real-world requirements, and translate them into well-structured technical specifications that ensure the system is both effective and user-friendly.

With the exponential growth of e-commerce, online customers are faced with thousands of products and fluctuating prices across multiple platforms. Traditional shopping often relies on impulsive decision-making or manual price comparisons, which are time-consuming and inefficient. As a result, there is an urgent need for a **centralized and automated solution** that can track price changes, notify users instantly of discounts, and provide insights about product trends.

Key Needs Identified:

1. **Transparency in Online Shopping**
    - Customers often do not know whether a product is genuinely discounted or if the "deal" is artificially created. Trendly provides complete transparency by showing historical data and price fluctuations.
    - Shoppers can verify if the current offer is truly beneficial or if waiting could provide a better deal.
2. **Smart and Informed Decision-Making**
    - Instead of relying on guesswork, users need concrete data to make purchasing decisions.
    - Trendly's **price history graphs** and analytics dashboard provide insights into seasonal discounts, demand-driven changes, and product popularity.
3. **Instant Price Drop Alerts**
    - Users cannot afford to monitor websites 24/7.
    - Automated **alerts via email (and later push/WhatsApp)** ensure they never miss out on an important price drop.
4. **User-Centric Shopping Experience**
    - Every user has a personal budget and target price in mind.
    - Trendly allows setting a **personalized watchlist and target price**, ensuring notifications are relevant and useful.
5. **Time and Effort Savings**
    - Manual price comparisons across multiple platforms are inefficient.
    - Trendly automates this entire process, saving time while improving shopping accuracy.

3B. FEASIBILITY STUDY

A feasibility study evaluates whether the Trendly system is practical, achievable, and sustainable in real-world conditions. It is not enough for a solution to be technically sound—it must also be economically viable, operationally convenient, legally compliant, and deliverable within realistic time constraints.

1. Technical Feasibility

- The MERN stack (MongoDB, Express.js, React.js, Node.js) provides a **scalable and reliable** base for development.
- Modern libraries such as **Chart.js** for data visualization, **Nodemailer** for email notifications, and **Node-cron** for scheduling ensure Trendly can deliver its promised functionality.
- Cloud platforms like AWS or MongoDB Atlas can easily scale the backend when the number of users grows.

2. Economic Feasibility

- Development costs remain moderate because the MERN stack and associated libraries are open-source.
- Hosting can begin on budget-friendly solutions like Heroku, Railway, or Vercel before scaling to enterprise-grade cloud infrastructure.
- Revenue potential exists in future through premium features (e.g., unlimited watchlist, AI-based predictions, ad-free version).

3. Operational Feasibility

- The system is designed with **intuitive UI/UX** using React and TailwindCSS, making it easy even for non-technical users.
- Since all major processes (price tracking, alert generation, analytics updates) are automated, the operational load is minimal.
- Admins can monitor data integrity and user accounts efficiently through backend tools.

4. Legal Feasibility

- Trendly must comply with **data privacy regulations** (GDPR, Indian IT Act).
- Only essential personal data (name, email, hashed password) is stored, ensuring compliance with security norms.
- Content scrapers must respect e-commerce websites' policies, meaning scraping needs to be carefully designed within legal boundaries.

5. Schedule Feasibility

- Development can be realistically achieved in **6–7 months**:
    - Month 1–2: Requirement gathering, UI/UX design.
    - Month 3–4: Backend & database development.
    - Month 5: Integration of price tracking, alerts, and analytics.
    - Month 6: Testing, deployment, and optimizations.

Overall Conclusion

The Trendly project is technically robust, economically viable, legally safe, and operationally efficient. It offers significant real-world value and can be implemented within a reasonable time frame and budget.

3C. WORKFLOW

The **System Development Life Cycle (SDLC)** plays a crucial role in Trendly's development because it ensures that each step is systematically planned and executed. For this project, the **Waterfall Model and Iterative Waterfall Model** were adopted to balance structure and flexibility.

Waterfall Model Design

- In the **traditional Waterfall Model**, the process is linear and sequential.
- Phases include Requirement Analysis → System Design → Implementation → Integration → Testing → Deployment → Maintenance.
- Each phase's output acts as the input for the next phase, making it simple and predictable.
- For Trendly, this model ensures that **requirements such as product tracking and alerts** are clearly defined before moving into development.

Iterative Waterfall Design

- The **Iterative Waterfall Model** is a more flexible variation where certain phases can be revisited.
- This allows the system to evolve with feedback while still maintaining structured documentation.
- In Trendly, the **price tracking engine and watchlist** were refined multiple times during development cycles for accuracy and user satisfaction.

Phases in Iterative Waterfall Model:

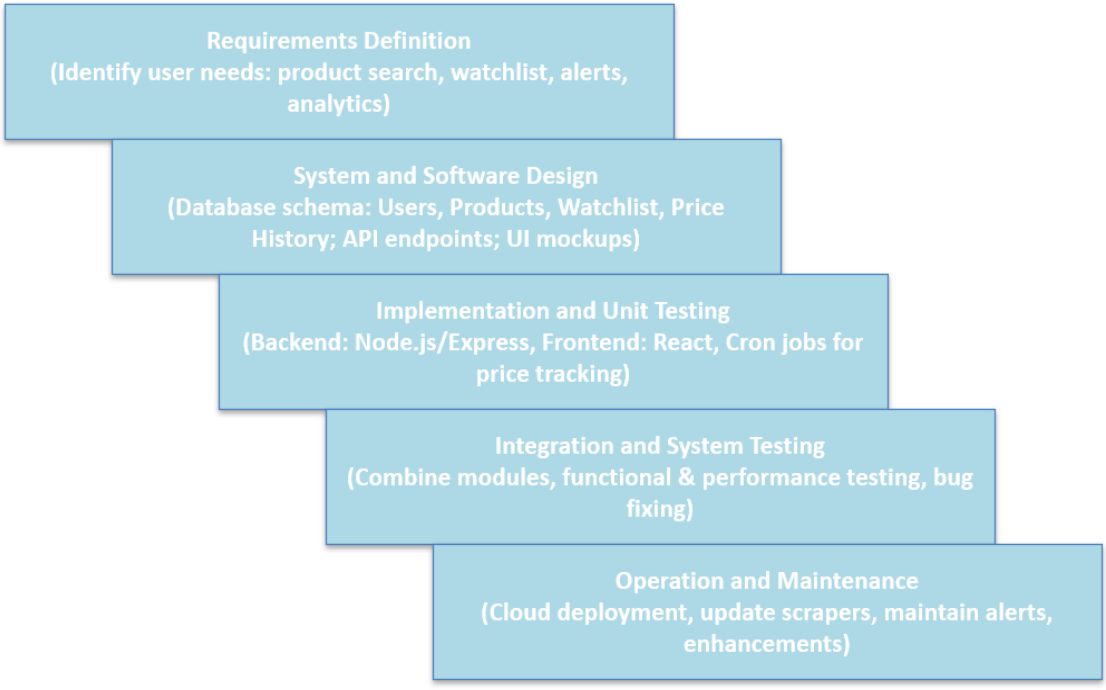
1. **Requirement Gathering & Analysis** – Identifying all functional needs like user login, watchlist, and price alerts.
2. **System Design** – Designing database schema (Users, Products, PriceHistory, Watchlist) and UI wireframes.
3. **Implementation** – Building backend APIs (Node.js, Express), frontend components (React, Tailwind), and cron jobs.
4. **Integration & Testing** – Merging modules, performing unit and system testing for accuracy.
5. **Deployment** – Hosting on a cloud server with continuous integration and monitoring tools.
6. **Maintenance** – Updating scrapers, improving alert mechanisms, adding new features like push notifications.

Advantages:

1. **Flexibility** – Requirements like multi-platform alerts can be added iteratively.
2. **Risk Management** – Issues are identified early during repeated cycles.
3. **Customer Feedback** – Early prototypes allow user testing and quick improvements.

Disadvantages:

1. **Higher Complexity** – Requires strong coordination between design, development, and testing.
2. **Scope Creep** – Frequent iterations may introduce unplanned features.
3. **Resource Intensive** – More time and cost if too many refinements occur.



▪ **Applications**

The **Iterative Waterfall Model** is particularly suitable for the development of **Trendly** because of the following reasons:

1. **Evolving User Requirements**
   - In an e-commerce price tracking system, user expectations (like additional filters, new alert methods such as WhatsApp/SMS, or enhanced analytics) evolve over time.
   - The iterative model allows refining these features without disturbing the entire workflow.
2. **Incremental Delivery of Core Features**
   - Trendly's modules (Authentication, Product Search, Watchlist, Price History, Alerts, Analytics Dashboard) can be developed and delivered in phases.
   - Stakeholders (test users/admins) can test each module independently and provide feedback.
3. **High Importance of Feedback Loops**
   - Since the system depends on real-time data from multiple e-commerce platforms, continuous **testing and adjustment of scrapers** is essential.
   - Iterative development ensures scrapers, cron jobs, and alert mechanisms can be improved quickly when site structures change.
4. **Risk Mitigation**
   - By breaking development into smaller cycles, risks such as **email deliverability issues, database scaling challenges, or UI performance bottlenecks** can be identified early and resolved.
5. **Practical Use Case for Trendly**
   - Initial version: focus on **basic search + watchlist + email alerts**.
   - Later iterations: add **analytics dashboards, advanced charts, push notifications, premium subscription model**.
   - This staged delivery ensures Trendly remains usable and valuable from the very first release.

3D. STUDY OF THE SYSTEM

The **Trendly Price Tracking Platform** is divided into several modules, each responsible for specific functionality. The modular design ensures scalability, maintainability, and ease of use. The modules of the system are described below:

● Register

1. **User Register**:
   - New users can register with their name, email, and password.
   - Passwords are encrypted using **bcrypt hashing** for security.
   - After registration, users can create a personal **watchlist** and set price-drop alerts.
2. **Admin Register (Future Scope)**:
   - Admins can be registered with elevated privileges.
   - Admins can manage products, monitor system performance, and oversee user activity.

● Login & Authentication

1. **User Login**:
   - Registered users log in using their email and password.
   - Authentication is handled using **JWT (JSON Web Tokens)**.
   - Sessions remain secure, and users gain access to their personal dashboard.
2. **Admin Login (Future Scope)**:
   - Admins can log in to manage product databases and view platform analytics.

● Product Search & Discovery

- A responsive **search bar with filters** (category, price range) allows users to find products.
- The system fetches product details including **name, image, price, category, and last updated date**.
- A **Trending Products** section highlights the most-watched items across the platform.

● Watchlist Management

- Users can add products to their personal **watchlist**.
- Each product entry in the watchlist includes:
    - Current price
    - Target price set by user
    - Alert toggle (Enable/Disable)
- Watchlists are persisted in the database and synced to the user's account.

● Price History Tracking

- Every tracked product has a **dedicated price history collection**.
- A **line chart (Chart.js)** displays price variations over time.
- Helps users make informed decisions about when to buy.

● Price Drop Alerts

- Implemented using **Node-cron scheduler** running at fixed intervals.
- When a product's price falls below the target price:
    - An **email notification** is sent to the user via **Nodemailer**.
    - Future upgrade: push notifications, WhatsApp alerts, SMS alerts.

● Analytics & Trends

- A **dashboard** provides insights into:
    - Trending product categories.
    - Most frequently watchlisted items.
    - Price drop patterns and frequency.
- Interactive **charts and graphs** allow users to analyze market trends in real-time.

● Account Management

1. **User Profile**:
    - View personal details, update preferences, and manage alerts.
    - Access saved watchlist and price history.
2. **Dashboard**:
    - Displays all watchlisted products.
    - Provides quick insights on active alerts and latest price drops.

3E. INPUT AND OUTPUT

The major inputs, outputs, and functions of the Trendly platform are summarized below:

INPUT

1. **User Registration Data**: Name, email, and password.
2. **Login Credentials**: Email & password for authentication.
3. **Product Search Query**: Keywords, filters (category, price range).
4. **Watchlist Parameters**: Product ID, target price, and alert toggle.

OUTPUT

1. **Product Information**: Current price, product image, details, and last updated timestamp.
2. **Price History Visualization**: Line chart showing past prices of a product.
3. **Notifications**:

    - Email alerts when a product's price drops.
    - Confirmation messages after adding/removing items from watchlist.
4. **Dashboard Analytics**: Reports on trending products, categories, and price changes.

3F. SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

The **Software Requirement Specification** defines the system requirements to ensure smooth development and deployment of the Trendly platform.

Developer Responsibilities

- **Developing** the system that meets all functional and non-functional requirements.
- **Demonstrating** the system and ensuring all modules work as expected.
- **Delivering documentation and user manual** for system usage.
- **Maintaining security** in data handling, especially in user authentication and notifications.

Functional Requirements

A. User Registration and Authentication

1. Users can securely create accounts with encrypted credentials.
2. System authenticates users via JWT and maintains secure login sessions.

B. Product Search and Discovery

1. Users can search for products with filters like category and price range.
2. Trending products are dynamically displayed based on user interest.

C. Watchlist & Alerts

1. Users can add/remove products to/from their watchlist.
2. Users can set custom target prices for price-drop notifications.
3. Email alerts are automatically triggered if the current price ≤ target price.

D. Price History Tracking

1. The system stores historical price data for each product.
2. Users can visualize price trends via an interactive chart.

E. Analytics Dashboard

1. Displays insights on most watched products and categories.
2. Shows frequency of price drops and buying trends.

Hardware Requirements

- Processor: Intel i3 or higher
- RAM: 8 GB minimum
- Storage: 20 GB SSD (for local development), scalable cloud storage for deployment
- Internet Connection: Required for product scraping and notifications

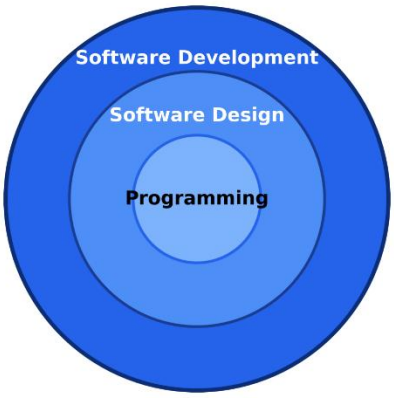
Software Requirements

- **Operating System**: Windows 11 / Linux (Ubuntu) / macOS

- **Frontend**: React.js, Redux Toolkit, TailwindCSS, Chart.js
- **Backend**: Node.js, Express.js, Node-cron, Nodemailer
- **Database**: MongoDB (MongoDB Atlas for cloud hosting)
- **Development Tools**:
    - Visual Studio Code (IDE)
    - Postman (API Testing)
    - Git & GitHub (Version Control)
- **Deployment**: Vercel/Netlify (Frontend), Heroku/Render (Backend), MongoDB Atlas (Database)

3G. SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms are the **systematic approaches** used to design, develop, and maintain software applications. For a complex solution like **Trendly**, which integrates **authentication, price scraping, data storage, analytics, and real-time notifications**, adopting robust software engineering paradigms ensures reliability, scalability, and maintainability.



1. Hierarchy of Paradigms

In the context of Trendly, the paradigms can be visualized as concentric layers:

- **Software Development Paradigm**
    - The broadest layer, covering the **entire lifecycle**: requirement gathering, system analysis, design, implementation, testing, deployment, and maintenance.
    - For Trendly, this includes **agile iteration cycles**, feature prioritization (alerts, analytics), and continuous integration with GitHub.
- **Software Design Paradigm**
    - A subset of development focusing on **architecture and system modeling**.
    - Trendly employs **modular architecture**: User Module, Watchlist Module, Price Tracker Module, Alert Module, Analytics Module.
    - Design tools used: **DFD, ER Diagrams, Sequence Diagrams** (already included in previous sections).

- **Programming Paradigm**

    - The most specific layer, dealing with **actual code implementation**.
    - Trendly leverages multiple programming paradigms:
        - **Object-Oriented Programming (OOP)** in Node.js (models, controllers, routes).
        - **Functional Programming** concepts in React (hooks, reducers).
        - **Event-driven programming** in Node-cron (scheduled jobs, asynchronous events).

2. Reliability in Trendly's Context

Reliability is critical because inaccurate price alerts could break user trust. Trendly ensures reliability at two levels:

1. **Requirement Reliability**
    - Ensured through detailed requirement gathering: secure login, product discovery, watchlist, alerts, analytics.
    - Iterative refinement ensures all **functional requirements** are captured.
2. **Operational Reliability**
    - Ensured by testing and continuous monitoring of modules:
        - **Authentication reliability**: JWT + bcrypt ensures secure sessions.
        - **Data reliability**: MongoDB with indexes ensures data consistency.
        - **Notification reliability**: Nodemailer + logging ensures emails are delivered.

3. Reliability Approaches Used in Trendly

1. **Error Avoidance**
    - Proper schema validation in MongoDB using Mongoose.
    - Input sanitization and API validation (e.g., no malformed product URLs).
    - Secure password hashing & token-based authentication.
2. **Error Detection and Correction**
    - Real-time validation: If a product URL is invalid or price parsing fails, the system throws descriptive errors.
    - Automated tests detect anomalies during builds (unit tests for APIs, React components).
    - Retry mechanisms in **price fetching & email sending**.
3. **Error Tolerance**
    - If one cron job fails (e.g., network error while fetching product), it retries in the next cycle without crashing the entire system.
    - Alerts are queued to prevent duplicate or missed notifications.
    - Degraded mode: Even if analytics fail, watchlist & alerts continue to function.

4. SYSTEM DESIGN

4A. DATA FLOW DIAGRAM (DFD)

A **Data Flow Diagram** models how information moves through Trendly, what processes transform it, which external entities interact with the system, and where data is stored. DFDs help stakeholders visualize the overall behavior without getting into timing or control-flow details.

Context (Level-0) Description

- **External_Entities:**
  User (shopper), **E-commerce Sites** (product pages being fetched), **Email Service** (SMTP provider).
- **Single Logical Process: Trendly System** — handles authentication, search, watchlist, tracking, alerts.
- **Data Store: MongoDB** (Users, Products, Watchlist, PriceHistory, Alerts).
- **Flows:**
    - *User → Trendly:* login/signup, search queries, add to watchlist, set target price.
    - *Trendly → User:* product results, price history charts, alert confirmations.
    - *Trendly → E-commerce Sites:* price fetch requests; *Sites → Trendly:* current price & metadata.
    - *Trendly → Email Service:* alert emails; *Email Service → Trendly:* delivery status.
    - *Trendly ↔ MongoDB:* read/write of users, products, watchlists, history, alerts.

Level-1 (Decomposition)

- **P1: Authentication & Account** – register/login (bcrypt + JWT), profile & preferences.

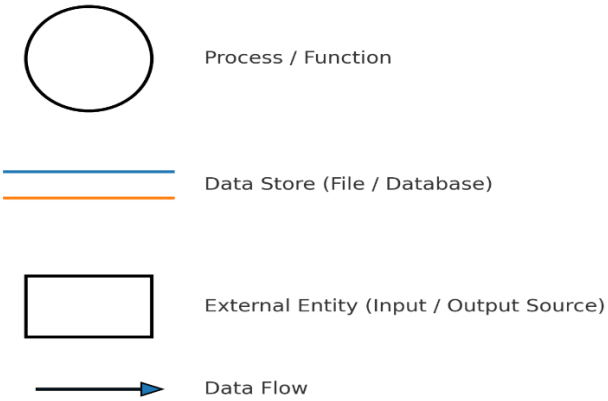
- **P2: Product Discovery** – search/filter products, show details & mini history.
- **P3: Watchlist & Targets** – add/remove items, set `targetPrice`, toggle `alertEnabled`.
- **P4: Price Tracker (Scheduler)** – periodic fetch, normalize, persist `PriceHistory`, evaluate triggers.
- **P5: Notifications** – compose email, send via SMTP, log status to `AlertsLog`.

DFD Notation (as used in diagrams)

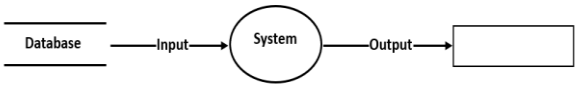
- **Process / Function:** circle (e.g., "Trendly System", "P2: Discovery").
- **Data Store:** double parallel lines (e.g., "MongoDB").
- **External Entity:** rectangle (e.g., "User", "E-commerce Sites").

Data Flow: directed arrow showing movement of data

DFD Notation:



DFD Example:



Steps to Construct Data Flow Diagram:

Four Steps are generally used to construct a DFD.

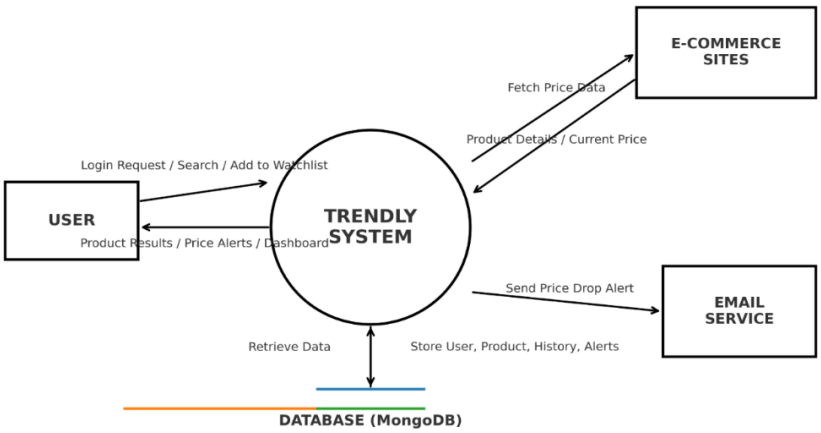
- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:

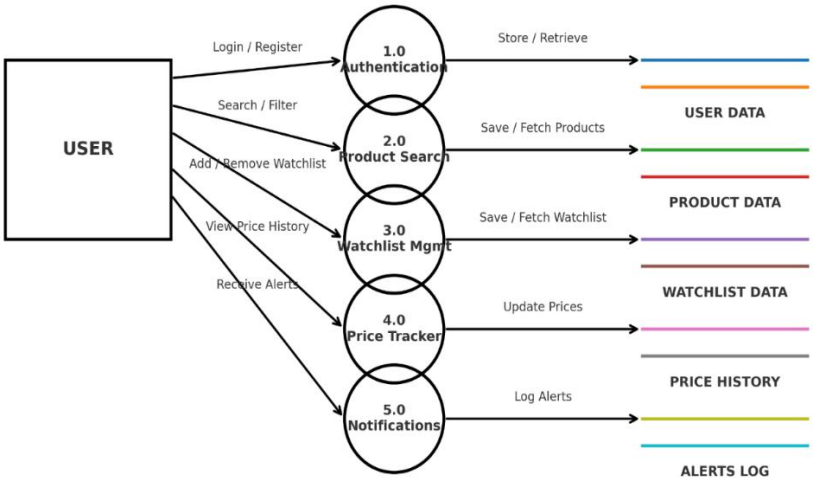
- Arrows should not cross each other.
- Squares, Circles, and Files must bear a name.
- Decomposed data flow squares and circles can have the same names.
- Draw all data flow around the outside of the diagram.

LEVEL 0 DFD OR CONTEXT DIAGRAM:

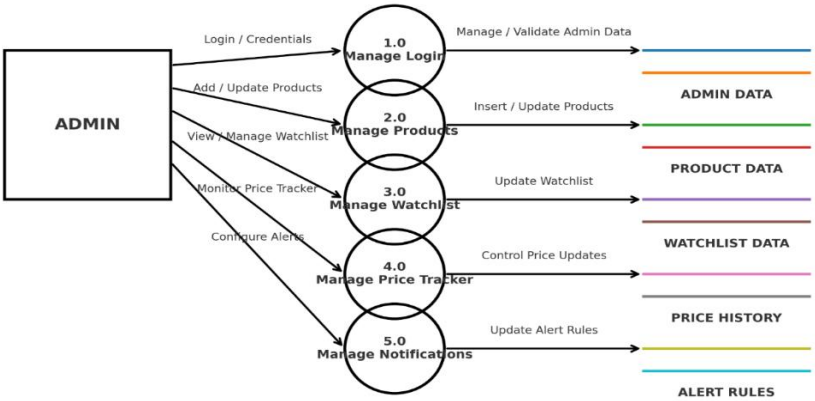
Login Reguest:



● LEVEL 1 DFD



● LEVEL 1 DFD

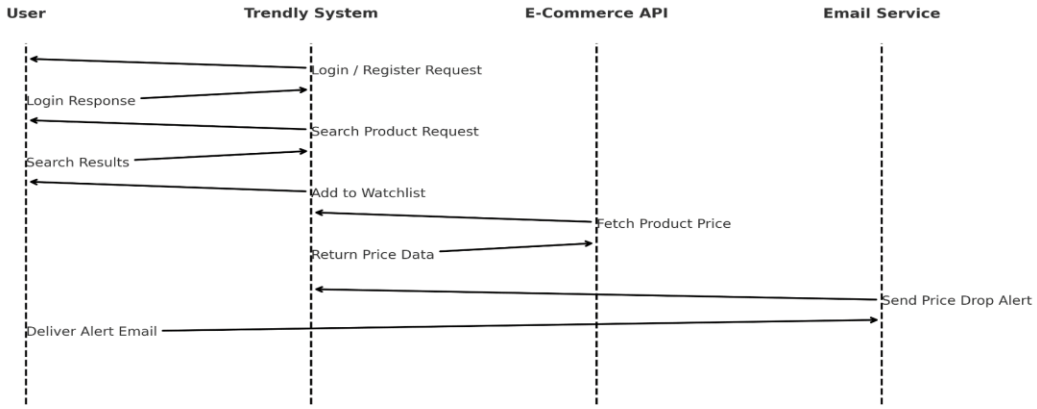


4B. SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that illustrates how processes in the **Trendly application** operate with one another and in what order. It is a construct of a Message Sequence Chart. In Trendly, the sequence diagram captures the flow of interactions between **User, Trendly System, E-commerce API, and Email Service**, showing how a price tracking and notification system functions step by step.

A sequence diagram shows object interactions arranged in a **time sequence**. It depicts the objects and components involved in the scenario and the sequence of messages exchanged between them to carry out the required functionality. For Trendly, this includes operations such as **user login, product search, watchlist management, price fetching, and alert notifications**.

Sequence diagrams are sometimes also referred to as **event diagrams** or **event scenarios**. They help in visualizing not just the static structure of the system but also the **dynamic behavior**, which is essential for understanding how the Trendly application interacts with external services and users in real-time.



Key Elements in Trendly's Sequence Diagram:

1. **Lifelines-(Parallel-Vertical-Lines):**
   Represent the different entities in Trendly that live simultaneously during system operation:
   - User
   - Trendly System (Core Application)
   - E-commerce API (for fetching real-time product price data)
   - Email Service (for sending alerts)

2. **Messages-(Horizontal-Arrows):**
   Represent the communication between these lifelines. In Trendly, key messages include:
   - Login / Register request and response
   - Product search queries and results
   - Adding/removing products from watchlist
   - Fetching price updates from E-commerce API
   - Sending price-drop alerts through Email Service
3. **Execution-Specification:**
   Each process is activated for a certain duration while handling requests (e.g., fetching data, verifying login, or sending an alert).

Dynamic Behavior in Trendly (Flow Description):

- The **User** initiates the process by logging into the Trendly platform. The **Trendly System** authenticates the credentials and returns a login confirmation.
- The **User** searches for a product. Trendly forwards the query to its database and if needed, fetches real-time prices from the **E-commerce API**, returning results to the user.
- The **User** adds a product to their **Watchlist**, setting a desired price threshold.
- Trendly's **Price Tracker** module communicates periodically with the **E-commerce API** to fetch updated price information.
- If the price drops below the target, the **Trendly System** triggers the **Email Service**, which sends a **Price Drop Alert** to the User.

Importance of Sequence Diagram:

- Helps to visualize how **internal modules** (authentication, search, watchlist, price tracker) and **external systems** (API, Email) interact.
- Provides a **dynamic view** of the application instead of just static structure.
- Makes it easier to **validate system requirements** and ensure that every functional requirement (login, search, alerting) is covered.
- Acts as a **blueprint for developers** to implement communication flows between frontend, backend, and external services.

How to draw Use Case Diagram?

Actors

1. **User** (online shopper)
2. **System** (Scheduler/Notifier that checks product prices & sends alerts)

Use Cases for User

- **Register**
- **Login**
- **Manage Profile** (update preferences, view watchlist)
- **Search Products**
- **View Product Details**
- **Add to Watchlist**
    - *Extends*: **Set Target Price**
    - *Extends*: **Enable/Disable Price Drop Alert**
- **View Price History** (chart view)
- **View Dashboard & Analytics** (trending items, categories, drops)
- **Logout**

Use Cases for System

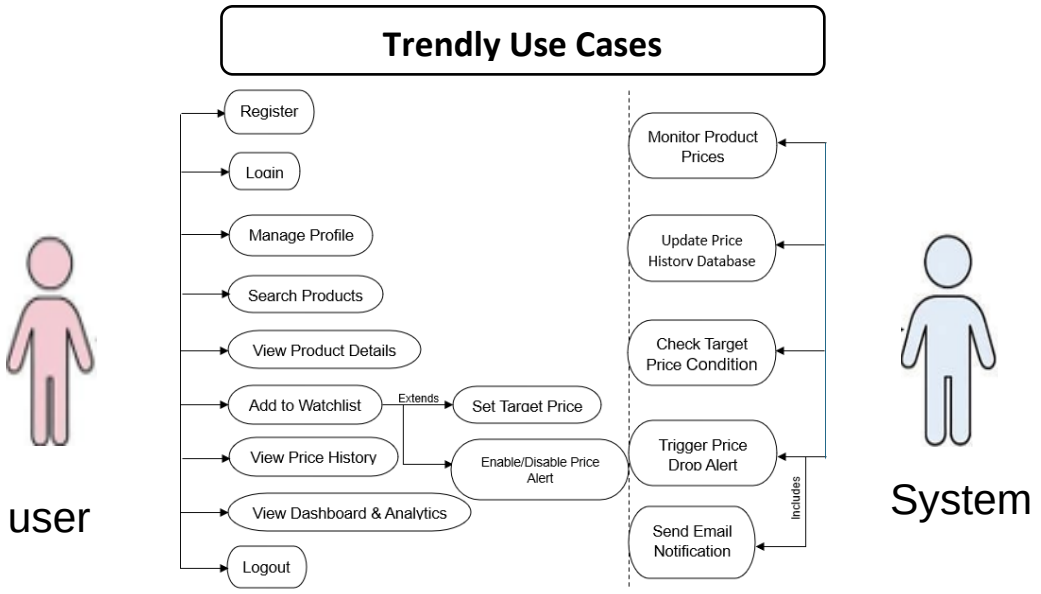
- **Monitor Product Prices** (via scraper / API)
- **Update Price History Database**
- **Check Target Price Condition**
- **Trigger Price Drop Alert**
    - *Includes*: **Send Email Notification**
    - *Future*: Push/WhatsApp Notifications

How it looks (description of diagram)

- Left side: **User** stick figure → connected to all above user use cases (register, login, watchlist, analytics).

- Right side: **System (Scheduler/Notifier)** stick figure → connected to price monitoring, database update, and alerts.
- Relationships:
  - *Add to Watchlist* → **extends** → *Set Target Price*
  - *Trigger Price Drop Alert* → **includes** → *Send Email*

USE CASE DIAGRAM

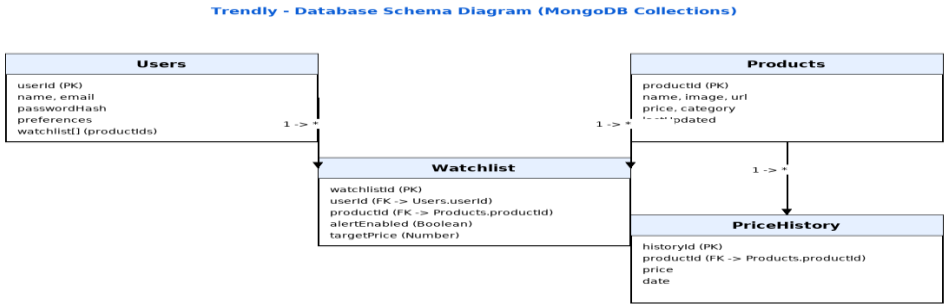


4D.SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to
create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

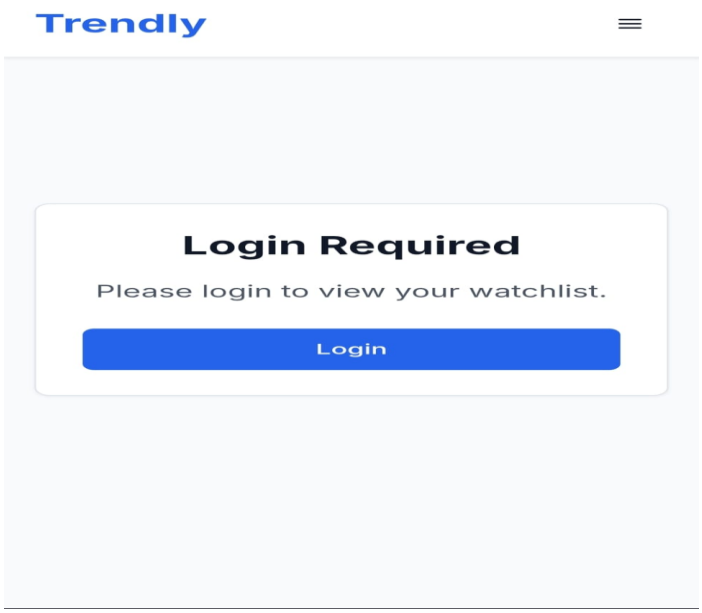
- **SCHEMA DESIGN:**

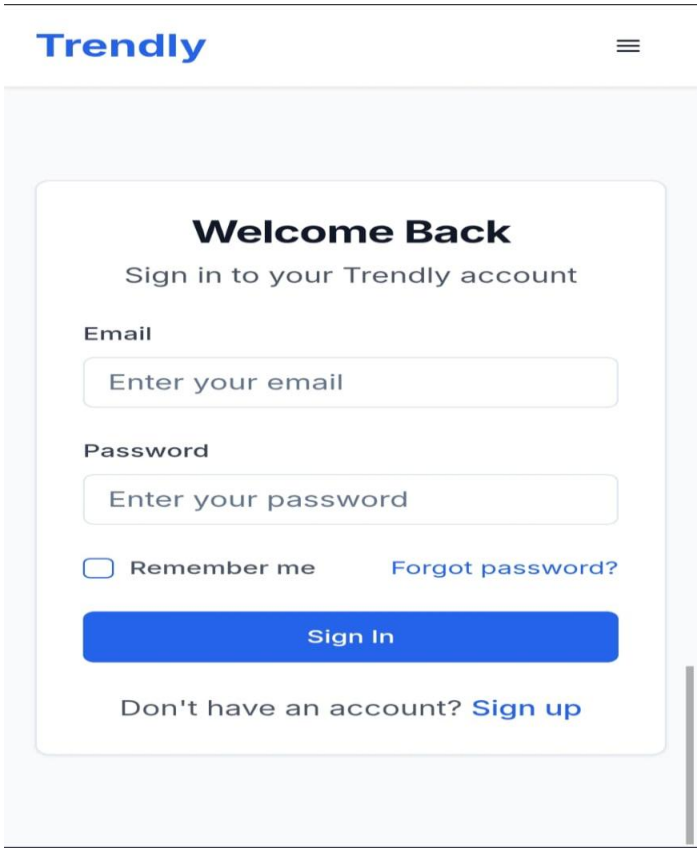


5. UI SNAPSHOT

❖ **FRONTEND : -**

- **Login Page:**

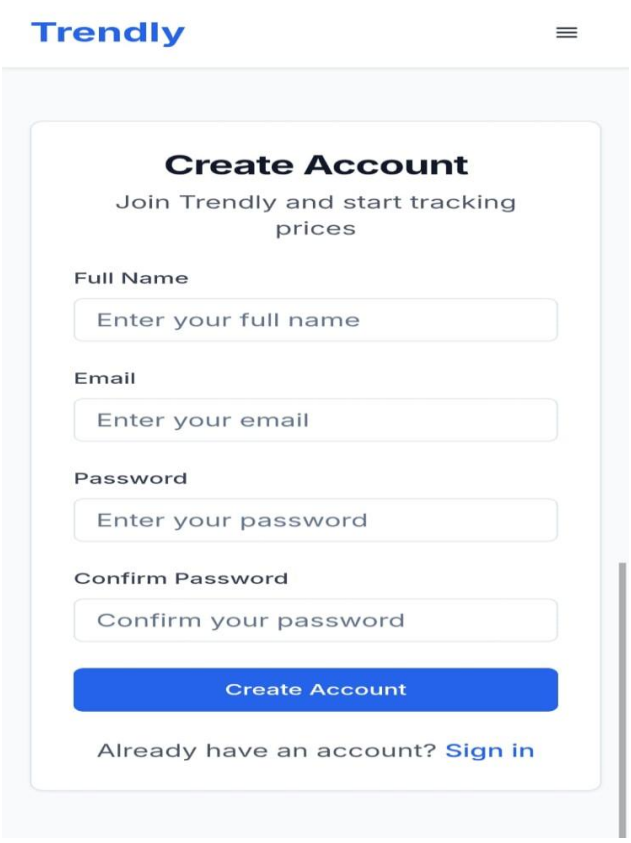




✓ **CODE**

```
import React from "react";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Card, CardContent } from "@/components/ui/card";
const SignIn = () => {
 return (
  <div className="p-6 bg-gray-50 min-h-screen flex items-center justify-center">
   <Card className="w-full max-w-md p-6">
    <CardContent>
     {/* Title */}
     <h1 className="text-2xl font-bold text-center mb-2">Welcome Back</h1>
     <p className="text-center text-gray-600 mb-6">
      Sign in to your Trendly account
     </p>
     {/* Form */}
     <form className="space-y-4">
      <div>
       <label className="block text-sm font-medium mb-1">Email</label>
       <Input type="email" placeholder="Enter your email" />
      </div>
      <div>
       <label className="block text-sm font-medium mb-1">Password</label>
       <Input type="password" placeholder="Enter your password" />
      </div>
      {/* Remember Me & Forgot Password */}
      <div className="flex items-center justify-between text-sm">
       <label className="flex items-center space-x-2">
        <input type="checkbox" className="h-4 w-4" />
        <span>Remember me</span>
       </label>
       <a href="/forgot-password" className="text-blue-600 font-medium">
        Forgot password?
       </a>
      </div>
      <Button className="w-full bg-blue-600 text-white">Sign In</Button>
     </form>
     {/* Footer */}
     <p className="text-center text-sm text-gray-600 mt-4">
      Don't have an account? {" "}
      <a href="/signup" className="text-blue-600 font-medium">
       Sign up
      </a>
     </p>
    </CardContent>
   </Card>
  </div>
 );};
export default SignIn;
```

Register Page:

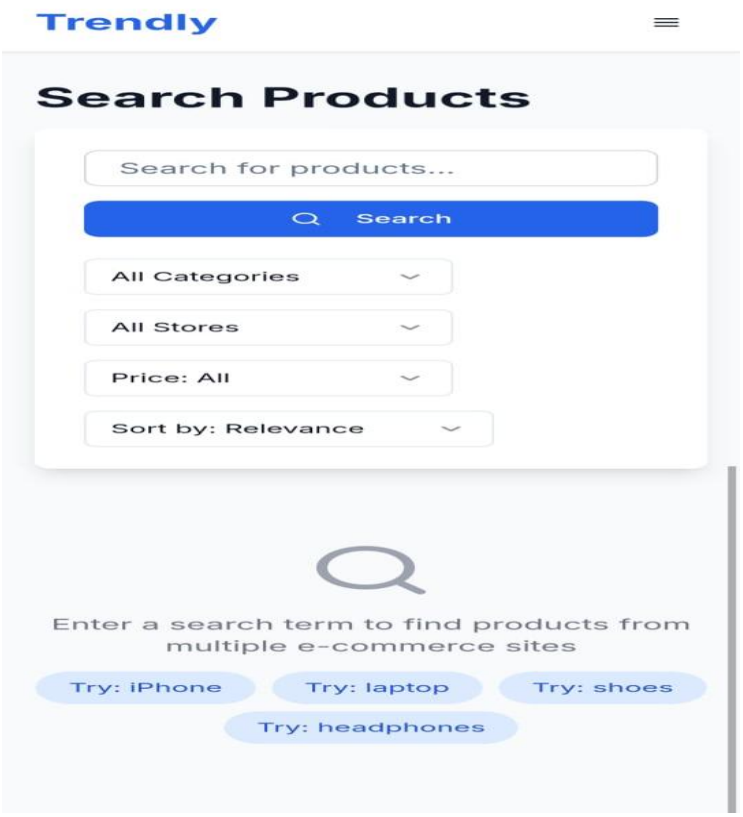


✔ CODE

```
import React from "react";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Card, CardContent } from "@/components/ui/card";
const CreateAccount = () => {
 return (
  <div className="p-6 bg-gray-50 min-h-screen flex items-center justify-center">
   <Card className="w-full max-w-md p-6">
    <CardContent>
     {/* Title */}
     <h1 className="text-2xl font-bold text-center mb-2">Create Account</h1>
     <p className="text-center text-gray-600 mb-6">
      Join Trendly and start tracking prices
     </p>
     {/* Form */}
     <form className="space-y-4">
      <div>
       <label className="block text-sm font-medium mb-1">Full Name</label>
       <Input placeholder="Enter your full name" />
      </div>
      <div>
       <label className="block text-sm font-medium mb-1">Email</label>
       <Input type="email" placeholder="Enter your email" />
      </div>
      <div>
       <label className="block text-sm font-medium mb-1">Password</label>
       <Input type="password" placeholder="Enter your password" />
```

```
          </div>
          <div>
            <label className="block text-sm font-medium mb-1">Confirm Password</label>
            <Input type="password" placeholder="Confirm your password" />
          </div>
          <Button className="w-full bg-blue-600 text-white">Create Account</Button>
        </form>
        {/* Footer */}
        <p className="text-center text-sm text-gray-600 mt-4">
          Already have an account? {" "}
          <a href="/signin" className="text-blue-600 font-medium">
            Sign in
          </a>
        </p>
      </CardContent>
    </Card>
  </div>
 );
};
export default CreateAccount;
```

3) Product Search Page



✔ CODE

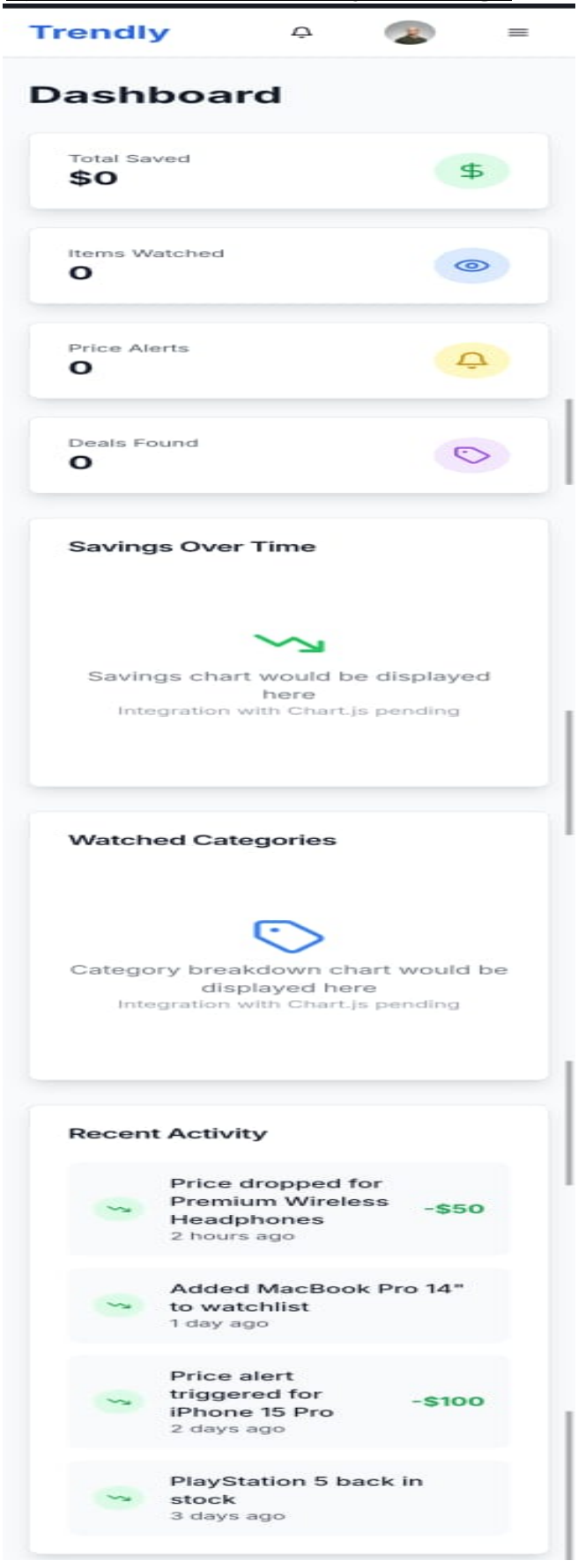
●components-pages-product.jsx

```
import React from "react";
```

```
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Card, CardContent } from "@/components/ui/card";
import { Select, SelectTrigger, SelectValue, SelectContent, SelectItem } from "@/components/ui/select";
import { Search } from "lucide-react";

const SearchProducts = () => {
 return (
  <div className="p-6 bg-gray-50 min-h-screen">
   {/* Header */}
   <h1 className="text-2xl font-bold mb-6">Search Products</h1>

   {/* Search Card */}
   <Card className="p-6 mb-10 max-w-lg">
    <CardContent className="space-y-4">
     {/* Search Bar */}
     <div className="flex space-x-2">
      <Input placeholder="Search for products..." className="flex-1" />
      <Button className="bg-blue-600 text-white flex items-center space-x-2">
       <Search className="w-4 h-4" />
       <span>Search</span>
      </Button>
     </div>

     {/* Filters */}
     <Select>
      <SelectTrigger>
       <SelectValue placeholder="All Categories" />
      </SelectTrigger>
      <SelectContent>
       <SelectItem value="all">All Categories</SelectItem>
       <SelectItem value="electronics">Electronics</SelectItem>
       <SelectItem value="fashion">Fashion</SelectItem>
      </SelectContent>
     </Select>

     <Select>
      <SelectTrigger>
       <SelectValue placeholder="All Stores" />
      </SelectTrigger>
      <SelectContent>
       <SelectItem value="all">All Stores</SelectItem>
       <SelectItem value="amazon">Amazon</SelectItem>
       <SelectItem value="flipkart">Flipkart</SelectItem>
      </SelectContent>
     </Select>

     <Select>
      <SelectTrigger>
       <SelectValue placeholder="Price: All" />
      </SelectTrigger>
      <SelectContent>
       <SelectItem value="all">All</SelectItem>
       <SelectItem value="low">Under $100</SelectItem>
       <SelectItem value="mid">$100 - $500</SelectItem>
       <SelectItem value="high">Above $500</SelectItem>
      </SelectContent>
```

```
        </Select>

        <Select>
          <SelectTrigger>
            <SelectValue placeholder="Sort by: Relevance" />
          </SelectTrigger>
          <SelectContent>
            <SelectItem value="relevance">Relevance</SelectItem>
            <SelectItem value="price_low">Price: Low to High</SelectItem>
            <SelectItem value="price_high">Price: High to Low</SelectItem>
            <SelectItem value="newest">Newest</SelectItem>
          </SelectContent>
        </Select>
      </CardContent>
    </Card>

    {/* Empty State */}
    <div className="text-center text-gray-500">
      <Search className="w-12 h-12 mx-auto mb-4 text-gray-400" />
      <p className="mb-4">Enter a search term to find products from multiple e-commerce sites</p>
      <div className="flex flex-wrap justify-center gap-3">
        {['iPhone', 'laptop', 'shoes', 'headphones'].map((item) => (
          <Button key={item} variant="outline" className="rounded-full px-4 py-1">
            Try: {item}
          </Button>
        ))}
      </div>
    </div>
  </div>
 );
};

export default SearchProducts;
```

4) Dashboard & Analytics Page

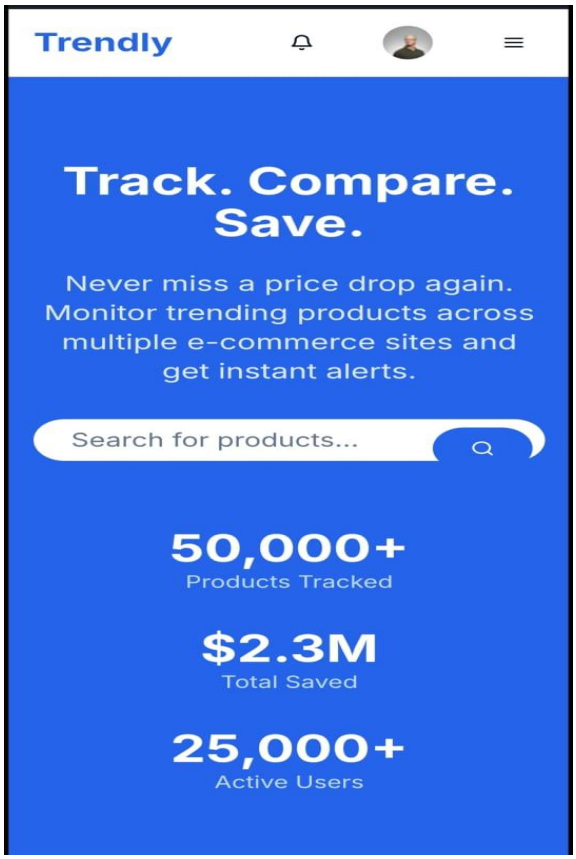


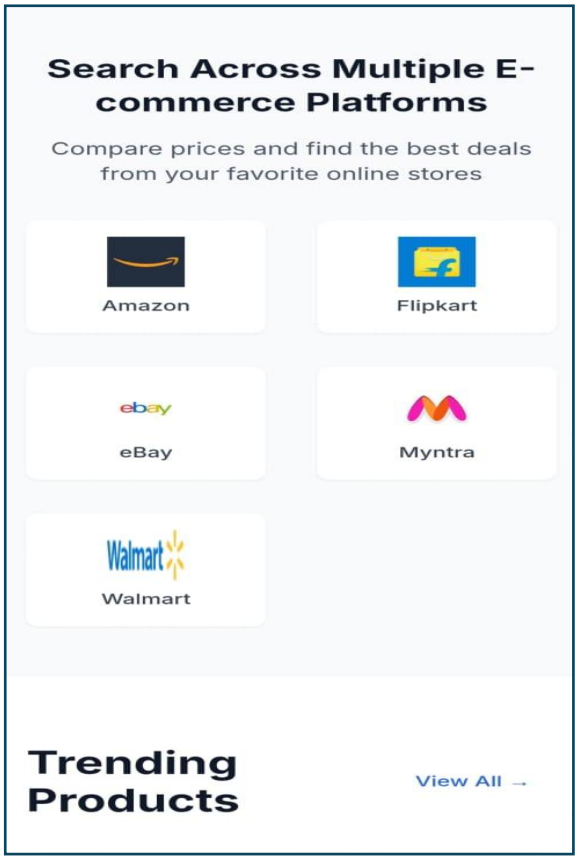
✔ CODE

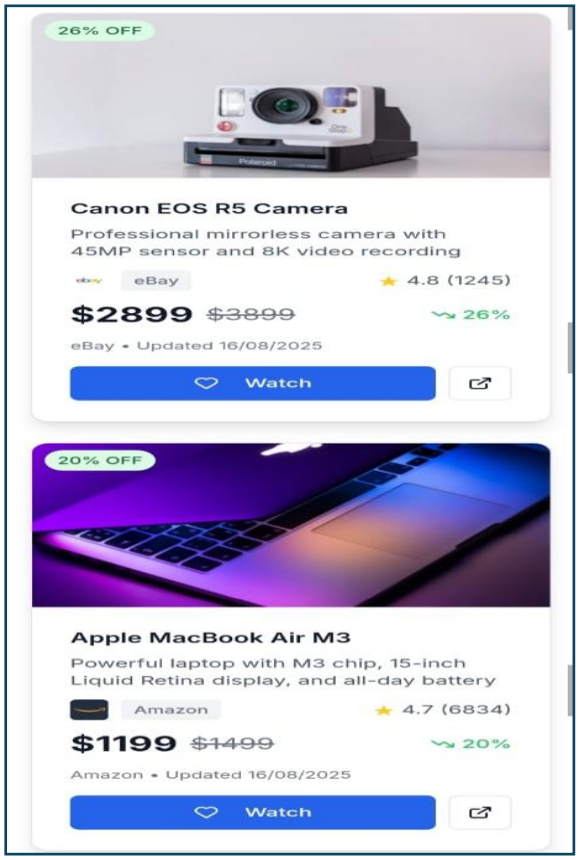
```
import React from "react";
import { Card, CardContent } from "@/components/ui/card";
import { Bell, Eye, Tag, DollarSign } from "lucide-react";

const Dashboard = () => {
  return (
    <div className="p-6 bg-gray-50 min-h-screen">
      {/* Header */}
      <header className="flex justify-between items-center mb-6">
        <h1 className="text-2xl font-bold">Dashboard</h1>
      </header>

      {/* Stats Section */}
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4 mb-6">
        <Card className="p-4 flex items-center justify-between">
          <div>
            <p className="text-gray-600">Total Saved</p>
            <h2 className="text-xl font-bold">$0</h2>
          </div>
          <DollarSign className="text-green-500 w-6 h-6" />
        </Card>

        <Card className="p-4 flex items-center justify-between">
          <div>
            <p className="text-gray-600">Items Watched</p>
            <h2 className="text-xl font-bold">0</h2>
          </div>
          <Eye className="text-blue-500 w-6 h-6" />
        </Card>

        <Card className="p-4 flex items-center justify-between">
          <div>
            <p className="text-gray-600">Price Alerts</p>
            <h2 className="text-xl font-bold">0</h2>
          </div>
          <Bell className="text-yellow-500 w-6 h-6" />
        </Card>

        <Card className="p-4 flex items-center justify-between">
          <div>
            <p className="text-gray-600">Deals Found</p>
            <h2 className="text-xl font-bold">0</h2>
          </div>
          <Tag className="text-purple-500 w-6 h-6" />
        </Card>
      </div>

      {/* Savings Over Time */}
      <Card className="mb-6">
        <CardContent className="p-6 text-center text-gray-500">
          <p className="mb-2 font-semibold">Savings Over Time</p>
          <p>Savings chart would be displayed here</p>
          <small>Integration with Chart.js pending</small>
        </CardContent>
```

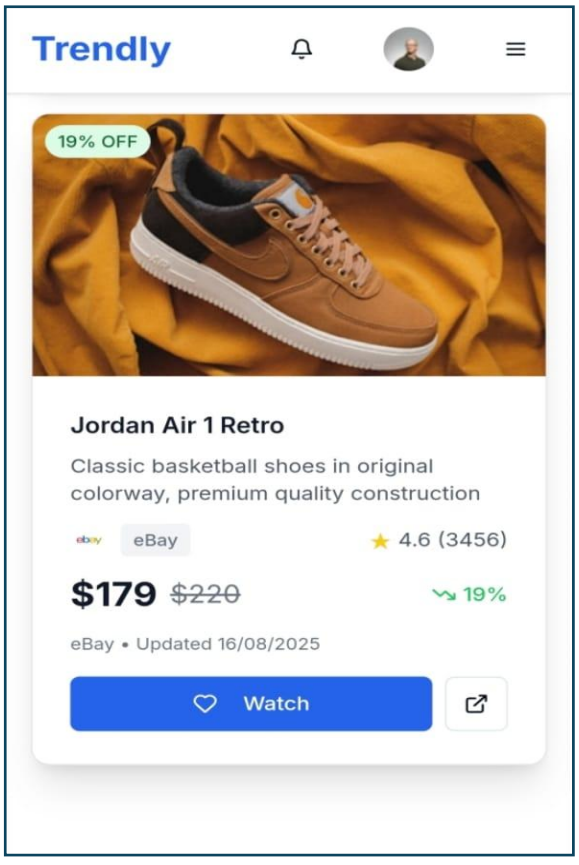
```
    </Card>

    {/* Watched Categories */}
    <Card className="mb-6">
      <CardContent className="p-6 text-center text-gray-500">
        <p className="mb-2 font-semibold">Watched Categories</p>
        <p>Category breakdown chart would be displayed here</p>
        <small>Integration with Chart.js pending</small>
      </CardContent>
    </Card>

    {/* Recent Activity */}
    <Card>
      <CardContent className="p-6">
        <h2 className="text-lg font-semibold mb-4">Recent Activity</h2>
        <div className="space-y-4">
          <div className="flex justify-between">
            <p>Price dropped for Premium Wireless Headphones</p>
            <span className="text-green-600 font-bold">- $50</span>
          </div>
          <p className="text-gray-500 text-sm">2 hours ago</p>

          <div>
            <p>Added MacBook Pro 14" to watchlist</p>
            <p className="text-gray-500 text-sm">1 day ago</p>
          </div>

          <div className="flex justify-between">
            <p>Price alert triggered for iPhone 15 Pro</p>
            <span className="text-green-600 font-bold">- $100</span>
          </div>
          <p className="text-gray-500 text-sm">2 days ago</p>

          <div>
            <p>PlayStation 5 back in stock</p>
            <p className="text-gray-500 text-sm">3 days ago</p>
          </div>
        </div>
      </CardContent>
    </Card>
  </div>
 );
};

export default Dashboard;
```

5) Home Page


Trendly
Track. Compare. Save.
Never miss a price drop again. Monitor trending products across multiple e-commerce sites and get instant alerts.
Search for products...
50,000+
Products Tracked
$2.3M
Total Saved
25,000+
Active Users


Search Across Multiple E-commerce Platforms
Compare prices and find the best deals from your favorite online stores
Amazon
Flipkart
eBay
Myntra
Walmart
Trending Products
View All →


26% OFF
Canon EOS R5 Camera
Professional mirrorless camera with 45MP sensor and 8K video recording
eBay
4.8 (1245)
$2899 $3899
26%
eBay • Updated 16/08/2025
Watch
20% OFF
Apple MacBook Air M3
Powerful laptop with M3 chip, 15-inch Liquid Retina display, and all-day battery
Amazon
4.7 (6834)
$1199 $1499
20%
Amazon • Updated 16/08/2025
Watch


Trendly
19% OFF
Jordan Air 1 Retro
Classic basketball shoes in original colorway, premium quality construction
eBay
4.6 (3456)
$179 $220
19%
eBay • Updated 16/08/2025
Watch

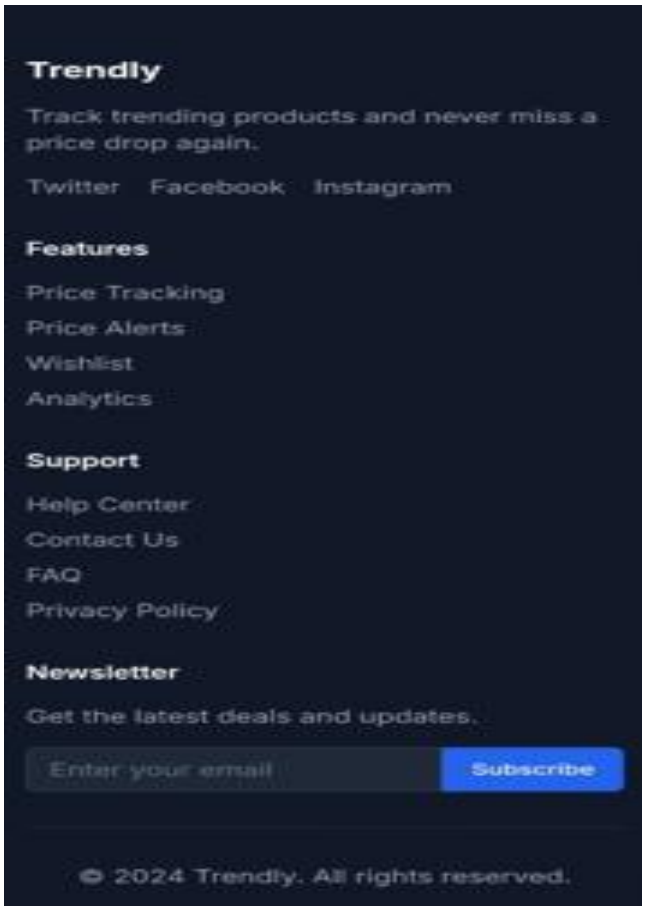
✔ CODE

```
import { useState } from "react";
import { Search, Bell, User, Menu } from "lucide-react";
import { Card, CardContent } from "@/components/ui/card";
import { Button } from "@/components/ui/button";

export default function TrendlyHome() {
 const [search, setSearch] = useState("");

 return (
  <div className="min-h-screen bg-white text-gray-900">
   {/* Header */}
   <header className="flex items-center justify-between px-4 py-3 shadow">
    <h1 className="text-xl font-bold text-blue-600">Trendly</h1>
    <div className="flex items-center gap-4">
     <Bell className="w-5 h-5 text-gray-600" />
     <User className="w-6 h-6 rounded-full bg-gray-300 p-1" />
     <Menu className="w-6 h-6 text-gray-700" />
    </div>
   </header>

   {/* Hero Section */}
   <section className="bg-blue-600 text-white text-center px-6 py-12 rounded-b-3xl">
    <h2 className="text-2xl font-bold">Track. Compare. Save.</h2>
    <p className="mt-2 text-sm opacity-90">
     Never miss a price drop again. Monitor trending products across
     multiple e-commerce sites and get instant alerts.
    </p>

    {/* Search Bar */}
    <div className="mt-6 flex items-center bg-white rounded-full overflow-hidden shadow w-full max-w-md
mx-auto">
     <input
      type="text"
      placeholder="Search for products..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
      className="flex-1 px-4 py-2 text-gray-800 outline-none"
     />
     <Button className="rounded-none rounded-r-full bg-blue-500 hover:bg-blue-700">
      <Search className="w-5 h-5 text-white" />
     </Button>
    </div>

    {/* Stats */}
    <div className="mt-8 flex justify-center gap-8">
     <div>
      <h3 className="text-xl font-bold">50,000+</h3>
      <p className="text-sm opacity-90">Products Tracked</p>
     </div>
     <div>
      <h3 className="text-xl font-bold">$2.3M</h3>
      <p className="text-sm opacity-90">Total Saved</p>
     </div>
     <div>
      <h3 className="text-xl font-bold">25,000+</h3>
```

```
        <p className="text-sm opacity-90">Active Users</p>
      </div>
    </div>
  </section>

  {/* Platforms */}
  <section className="px-6 py-10">
    <h3 className="text-lg font-semibold text-center">
      Search Across Multiple E-commerce Platforms
    </h3>
    <p className="text-sm text-center text-gray-500 mt-1">
      Compare prices and find the best deals from your favorite online stores
    </p>

    <div className="grid grid-cols-3 gap-6 mt-6">
      {[
        { name: "Amazon", logo: "https://cdn-icons-png.flaticon.com/512/732/732228.png" },
        { name: "Flipkart", logo: "https://cdn-icons-png.flaticon.com/512/6124/6124990.png" },
        { name: "eBay", logo: "https://cdn-icons-png.flaticon.com/512/888/888879.png" },
        { name: "Myntra", logo: "https://cdn-icons-png.flaticon.com/512/5968/5968841.png" },
        { name: "Walmart", logo: "https://cdn-icons-png.flaticon.com/512/731/731985.png" },
      ].map((site, i) => (
        <Card key={i} className="flex items-center justify-center p-6 shadow hover:shadow-lg">
          <img src={site.logo} alt={site.name} className="w-10 h-10 mb-2" />
          <p className="text-sm font-medium">{site.name}</p>
        </Card>
      ))}
    </div>
  </section>

  {/* Trending Products */}
  <section className="px-6 pb-12">
    <div className="flex justify-between items-center mb-4">
      <h3 className="text-lg font-semibold">Trending Products</h3>
      <a href="#" className="text-sm text-blue-600 font-medium">
        View All →
      </a>
    </div>

    <div className="flex flex-col gap-6">
      {/* Product 1 */}
      <Card className="overflow-hidden shadow-md">
        <CardContent className="p-4">
          <div className="relative">
            <span className="absolute top-2 left-2 bg-green-100 text-green-600 text-xs px-2 py-1 rounded-md
font-medium">
              26% OFF
            </span>
            <img
              src="https://cdn.pixabay.com/photo/2020/05/22/18/18/camera-5203189_1280.jpg"
              alt="Canon EOS R5"
              className="rounded-lg mb-3"
            />
          </div>
          <h4 className="font-semibold">Canon EOS R5 Camera</h4>
          <p className="text-sm text-gray-500">
            Professional mirrorless camera with 45MP sensor and 8K video recording
```

```
          </p>
          <p className="mt-2 font-bold text-lg">$2899 <span className="line-through text-gray-400 text-
sm">$3809</span></p>
          <p className="text-xs text-gray-500">eBay • Updated 16/08/2025</p>
          <Button className="mt-3 w-full bg-blue-600 hover:bg-blue-700">Watch</Button>
         </CardContent>
        </Card>

        {/* Product 2 */}
        <Card className="overflow-hidden shadow-md">
         <CardContent className="p-4">
          <div className="relative">
           <span className="absolute top-2 left-2 bg-green-100 text-green-600 text-xs px-2 py-1 rounded-md
font-medium">
            20% OFF
           </span>
           <img
            src="https://cdn.pixabay.com/photo/2017/07/31/11/21/apple-2557572_1280.jpg"
            alt="MacBook Air M3"
            className="rounded-lg mb-3"
           />
          </div>
          <h4 className="font-semibold">Apple MacBook Air M3</h4>
          <p className="text-sm text-gray-500">
           Powerful laptop with M3 chip, 15-inch Liquid Retina display, and all-day battery
          </p>
          <p className="mt-2 font-bold text-lg">$1199 <span className="line-through text-gray-400 text-
sm">$1499</span></p>
          <p className="text-xs text-gray-500">Amazon • Updated 16/08/2025</p>
          <Button className="mt-3 w-full bg-blue-600 hover:bg-blue-700">Watch</Button>
         </CardContent>
        </Card>
       </div>
     </section>
    </div>
  );
}
```

6) About Page:



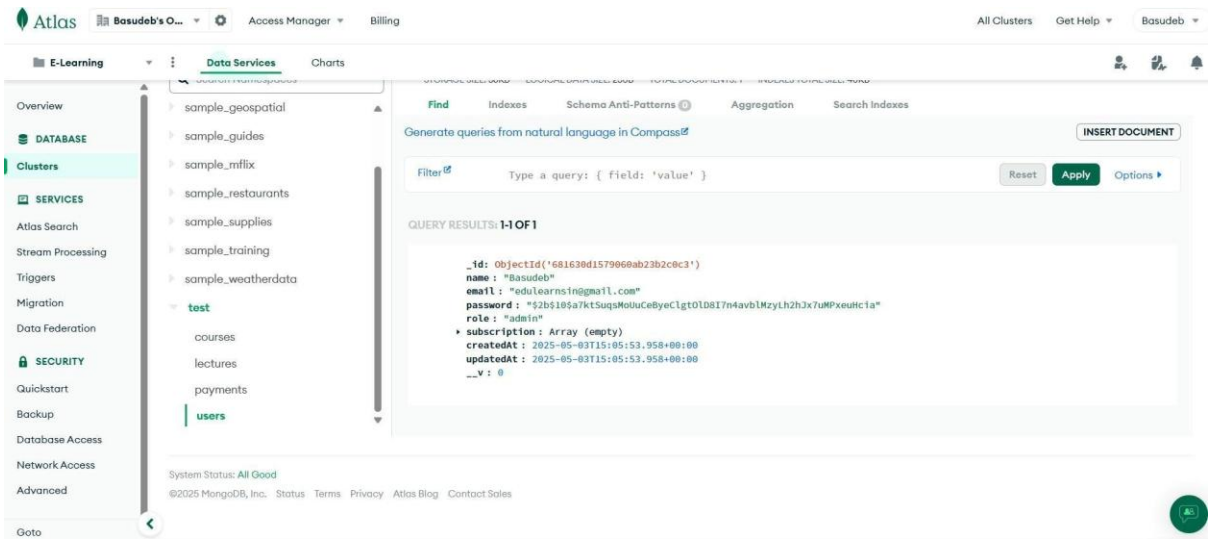
✔ CODE

```
import { useState } from "react";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";

export default function Footer() {
 const [email, setEmail] = useState("");

 return (
  <footer className="bg-[#0A0F1C] text-gray-300 py-10 px-6">
   <div className="max-w-6xl mx-auto grid grid-cols-1 md:grid-cols-4 gap-8">
    {/* Brand Section */}
    <div>
     <h2 className="text-lg font-semibold text-white">Trendly</h2>
     <p className="mt-2 text-sm text-gray-400">
      Track trending products and never miss a price drop again.
     </p>
     <div className="flex gap-4 mt-4 text-sm text-gray-400">
      <a href="#" className="hover:text-white">Twitter</a>
      <a href="#" className="hover:text-white">Facebook</a>
      <a href="#" className="hover:text-white">Instagram</a>
     </div>
    </div>
```

```
      {/* Features */}
      <div>
       <h3 className="text-sm font-semibold text-white mb-3">Features</h3>
       <ul className="space-y-2 text-sm">
        <li><a href="#" className="hover:text-white">Price Tracking</a></li>
        <li><a href="#" className="hover:text-white">Price Alerts</a></li>
        <li><a href="#" className="hover:text-white">Wishlist</a></li>
        <li><a href="#" className="hover:text-white">Analytics</a></li>
       </ul>
      </div>

      {/* Support */}
      <div>
       <h3 className="text-sm font-semibold text-white mb-3">Support</h3>
       <ul className="space-y-2 text-sm">
        <li><a href="#" className="hover:text-white">Help Center</a></li>
        <li><a href="#" className="hover:text-white">Contact Us</a></li>
        <li><a href="#" className="hover:text-white">FAQ</a></li>
        <li><a href="#" className="hover:text-white">Privacy Policy</a></li>
       </ul>
      </div>

      {/* Newsletter */}
      <div>
       <h3 className="text-sm font-semibold text-white mb-3">Newsletter</h3>
       <p className="text-sm text-gray-400 mb-3">
        Get the latest deals and updates.
       </p>
       <div className="flex gap-2">
        <Input
         type="email"
         placeholder="Enter your email"
         value={email}
         onChange={(e) => setEmail(e.target.value)}
         className="bg-gray-800 border-gray-700 text-gray-200 placeholder-gray-500"
        />
        <Button className="bg-blue-600 hover:bg-blue-700 text-white">Subscribe</Button>
       </div>
      </div>
     </div>

     {/* Footer Bottom */}
     <div className="mt-10 border-t border-gray-800 pt-4 text-center text-sm text-gray-500">
      © 2024 Trendly. All rights reserved.
     </div>
    </footer>
  );
}
```

❖ BACKEND:-

1) USER AND ADMIN DATA:



✔ **CODE**

- **Database - db.js:**

```
import mongoose from "mongoose";
 export const connectDB = async () => { try
{ await mongoose.connect(process.env.DB); console.log("Database Connected");
} catch (error) { console.log(error);
}
};
```

2) model – User.js

✔ **CODE**

```
import mongoose, { Schema } from "mongoose";

const userSchema = new Schema(
 {
   name: { type: String, required: true, trim: true, maxlength: 120 },
   email: { type: String, required: true, unique: true, lowercase: true, index: true },
   passwordHash: { type: String, required: true }, // store hash, not raw password
```

```
    avatarUrl: { type: String },
    roles: { type: [String], default: ["user"] },
  },
  { timestamps: true }
);


userSchema.methods.toSafeJSON = function () {
  const obj = this.toObject();
  delete obj.passwordHash;
  return obj;
};
export const User = mongoose.models.User || mongoose.model("User", userSchema);
```

● model –Product.js

✔ **CODE**

```
import mongoose, { Schema } from "mongoose";

const productSchema = new Schema(
  {
    title: { type: String, required: true, trim: true },
    url: { type: String, required: true, unique: true },
    image: { type: String },
    store: { type: String, required: true, index: true }, // e.g., Amazon, Flipkart
    category: { type: String, index: true },
    currency: { type: String, default: "USD" },
    currentPrice: { type: Number, required: true, min: 0 },
    lastCheckedAt: { type: Date, default: Date.now },
    // denormalized stats
    watchersCount: { type: Number, default: 0, index: true },
  },
  { timestamps: true }
);

productSchema.index({ title: "text", store: 1, category: 1 });

export const Product = mongoose.models.Product || mongoose.model("Product", productSchema);
```

6. CONCLUSION

The **Trendly application** emerges as a robust and innovative solution for online shoppers who wish to make smarter and more informed purchasing decisions. By integrating features such as **price history tracking, watchlist management, and automated price drop alerts**, the system provides users with real-time insights into product trends and pricing fluctuations.

Through its **intuitive interface and powerful analytics dashboard**, Trendly empowers users to identify the right time to purchase, avoid overspending, and stay updated with market trends. The inclusion of **email alerts, target price notifications, and visual price charts** enhances the overall shopping experience by combining convenience with transparency.

As e-commerce continues to expand rapidly, **Trendly bridges the gap between shoppers and fluctuating product markets**, ensuring that users are always in control of their buying choices. With future enhancements such as push notifications and third-party integrations, the application holds immense potential to become an essential tool for every online shopper, contributing to **smarter, data-driven, and cost-effective shopping**.

7. FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future Scope:

The future of **Trendly** is highly promising with vast opportunities to expand features and improve user experience. Key areas for development include:

- **AI-Powered Recommendations:** Using machine learning to suggest products based on user preferences, shopping behavior, and market trends.
- **Smart Price Predictions:** Forecasting future price changes using historical data and predictive analytics to help users decide the best time to purchase.
- **Push Notifications & Multi-Channel Alerts:** Expanding beyond emails to include **mobile push alerts, SMS, and WhatsApp notifications** for instant updates.
- **Browser Extensions & Mobile App:** Providing users with quick access to product tracking directly from e-commerce websites or on-the-go.
- **Multi-Market Support:** Expanding tracking to cover multiple e-commerce platforms globally (Amazon, Flipkart, eBay, etc.) to cater to a wider audience.
- **Social & Collaborative Shopping:** Allowing users to share watchlists, compare product prices with friends, and follow trending products collectively.

❖ Further Enhancements:

1. User Experience Enhancements:

- **Personalized Dashboards:** Display insights like price trends, active watchlists, and personalized product suggestions.
- **Gamification:** Introduce badges, streaks, or achievements for users who actively track and save money.
- **Dark Mode & Accessibility Features:** Ensure usability for all users, including visually impaired shoppers.

2. Product & Data Features:

- **Advanced Filters & Search:** Category-based filtering, smart search, and AI-powered suggestions.
- **Price Comparison Tools:** Allow users to compare prices of the same product across multiple e-commerce platforms.
- **Auto-Update Product Info:** Keep product details, reviews, and images updated through automated scrapers.

3. *Community & Social Features:*

- **User Reviews & Feedback:** Let users share insights about price drops and experiences.
- **Group Watchlists:** Enable collaborative shopping by letting groups track the same product.
- **Community Trends:** Showcase popular watchlisted products and trending categories globally.

4. *Analytics & Insights:*

- **Detailed Price Reports:** Show historical price changes, drop frequency, and future predictions.
- **Spending Analytics:** Help users understand how much they saved using Trendly.
- **Smart Alerts:** Differentiate between small fluctuations and significant price drops.

5. *Tech Integrations:*

- **Cloud Sync & Offline Access:** Allow users to access their watchlist and data anytime.
- **Multi-Language Support:** Expand globally with multilingual dashboards and notifications.
- **Integration with Third-Party Tools:** Sync with shopping assistants, payment apps, and e-wallets.

✨ With these future developments, **Trendly** can evolve from a simple price tracker into a **complete smart shopping assistant**, enabling users to save money, make data-driven purchasing decisions, and enjoy a highly personalized shopping experience.

8. BIBLIOGRAPHY

1. www.w3schools.com – For web development references (HTML, CSS, JavaScript, React).
2. www.youtube.com – For tutorials and practical implementation guides.
3. www.pexels.com – For free stock images used in UI design.
4. www.codepen.io – For testing and experimenting with frontend components.
5. www.google.com – For research and resources on MERN stack development.
6. https://fonts.google.com fonts.google.com – For integrating modern and readable fonts.
7. https://reactjs.org reactjs.org – Official documentation of React.js used in frontend development.
8. www.mongodb.com – For database schema and query references in MongoDB.
9. https://expressjs.com expressjs.com – Official documentation of Express.js for backend API development.
10. https://nodejs.org nodejs.org – Official documentation of Node.js for server-side programming.