Prompts

My Prompt:

create a very detailed document for me with below details:
Prompt to Generate Comprehensive SDLC Documentation for NeighBorrow Using AI
Objective: Create a full suite of SDLC artifacts for the NeighBorrow platform, as outlined in the provided project plan, using AI tools. Ensure alignment with the defined purpose, features, tech stack, and timelines.

1. Project Planning
   Generate a Gantt chart or timeline reflecting the 3-month schedule, including tasks like UI design, authentication setup, AI integration, and testing.

Refine the budget breakdown (as in Section 9) into a visual format (e.g., pie chart) using tools like Excel or Figma.

Create a risk management matrix expanding Section 10, detailing risks (e.g., AI delays, low adoption), likelihood, impact, and mitigation steps.

2. Requirements Gathering
   Convert functional and non-functional requirements (Section 4) into:

User stories (e.g., “As a user, I want to filter items by location to find nearby resources”).

MoSCoW Prioritization (Must-have, Should-have, Could-have, Won’t-have) for post-MVP features.

Draft a Software Requirements Specification (SRS) document.

3. Use Case Definition
   Create use case diagrams (tools: Lucidchart, Draw.io) for core features:

Item sharing (list, search, recommend).

Skill exchange (barter, matching).

Event posting and notifications.

Include scenarios (e.g., “User A borrows a ladder from User B”).

4. System and UI Design
   System Architecture:

Generate a diagram (using Mermaid.js) for the high-level architecture (frontend, backend, AI layer, database).

Expand the database schema (Section 7) into Entity-Relationship Diagrams (ERDs).

UI/UX Design:

Create Figma wireframes for key pages: registration, item listing, messaging, and event board.

Ensure alignment with responsive design (Section 4) and Tailwind CSS styling.

5. User Personas
   Develop 4 detailed personas based on Section 2 (target audience):

Example: *“Sarah, 32, suburban parent seeking cost-saving tools and local events.”*

Include demographics, goals, pain points, and platform usage scenarios.

6. Code Implementation
   Generate boilerplate code for:

React.js frontend (e.g., item search component with filters).

Node.js/Express.js API endpoints (e.g., /api/items/search).

MongoDB schema definitions (Users, Items, Transactions).

AI Integration:

Python snippets for recommendation engines (collaborative filtering).

Example: “Integrate OpenAI API for chatbot responses.”

7. Test Scripts
   Create test cases for core functionalities:

Authentication (success/failure scenarios).

Item search (filters, AI recommendations).

Transaction flow (borrowing, review submission).

Generate automation scripts (e.g., Jest for React, Postman for API testing).

8. Documentation
   Technical Documentation:

API documentation (Swagger/OpenAPI).

Deployment guide (AWS/Heroku setup).

User Manual:

Step-by-step guides for listing items, exchanging skills, and posting events.

9. Contracts and Legal Artifacts
   Draft template agreements:

Terms of Service (ToS) and Privacy Policy (GDPR/CCPA compliance).

Service Level Agreement (SLA) for hosting uptime.

Risk Assessment Report: Expand Section 10 with compliance checklists.

10. Additional Artifacts
    Marketing Plan: Outline strategies for user acquisition (e.g., social media, community partnerships).

Analytics Dashboard: Mockup for tracking success metrics (Section 11: active users, retention).

Output Format:

Deliverables in Markdown or PDF with clear sections, diagrams, and code blocks.

Use AI Tools: ChatGPT for text, DALL-E for diagrams, GitHub Copilot for code, and Figma AI for wireframes.

Final Note: Ensure all components align with the original NeighBorrow vision (community-driven, AI-enhanced sharing) and adhere to the budget and timeline constraints.

refer to my software planning document and create a well detailed document for me with all the details.

My Prompt:

Here is my website and idea make a web-based application using the tech stack mentioned:

1. Definition of Purpose
Neigh Borrow is a web-based platform designed to foster community engagement
by enabling neighbors to share resources, skills, and services in a secure and user-
friendly manner. The platform will use AI to enhance user experience through
personalized recommendations, trust-building, and efficient matching.
2. Target Audience
• Primary Users: Residents of local communities in urban and suburban areas in
the USA.
• Demographics: Adults aged 25–55 who are tech-savvy and interested in community building, sustainability, and cost-saving.
• Secondary Users: Local businesses, community organizations, and event organizers who want to engage with the community.
=====
Frontend
• Framework: React.js (for responsive and dynamic UI).
• Styling: Tailwind CSS.
Backend
• Framework: Node.js with Express.js (lightweight and scalable).
• Database: MongoDB (for flexible and scalable data storage).
• AI/ML: Python with TensorFlow/PyTorch for AI models, or pre-built APIs like

OpenAI for NLP tasks.
Cloud Infrastructure
• Hosting: AWS (EC2 for backend, S3 for storage) or Heroku (for simplicity).
• Authentication: Firebase Authentication or Auth0.
• Real-time Communication: Socket.io for messaging.
AI Tools
• Recommendation Engine: Collaborative filtering or content-based filtering.
• Fraud Detection: Pre-built APIs like Google Cloud AI or custom models.
• Chatbot: OpenAI GPT or Dialogflow.
Core Features
1. Item Sharing:
o Users can list items to lend or borrow (e.g., tools, equipment, bikes).
o Search and filter items by category, location, and availability.
o AI-powered recommendations for items based on user preferences and past activity.
2. Skill Exchange:
o Users can offer or request services (e.g., tutoring, pet-sitting, gardening).
o Barter system for exchanging skills or items.
o AI-based matching of users with complementary needs and skills.
3. Local Events and Alerts:
o A bulletin board for posting community events, alerts, and announcements.
o AI-driven event recommendations based on user interests and location.
4. Trust and Reviews:
o User profiles with ratings and reviews.
o AI-powered fraud detection and trust scoring to ensure safety.
Functional Requirements
• User registration and authentication (email, phone, or social login).
• Item/service listing and search functionality.
• Messaging system for communication between users.
• Review and rating system.
• Event posting and notification system.
• AI-based recommendations and matching.
Non-Functional Requirements
• Scalability to handle 10,000+ users.
• Secure data storage and transmission (GDPR and CCPA compliance).
• Responsive design for mobile and web.
• Fast and reliable performance (under 2-second load time).

=====
ignore aws and cloud based app for hosting

My Prompt:

can you make the sign in button work then it should go to sign in page and it should show already a neighborrower if not then sign up and it should go to sign up page.

My Prompt:

currently this application logins user for any user name but it should sign in users with proper credentials only like a proper email and 8 digit password minimum and max 64 characters if this criteris doesn't meet then an error message should pop up.

My Prompt:

now in the sign up form ask for user location as well and in the location field you can suggest auto fill location option as the user types their location and there should be an option to auto detect location.

My Prompt:

upon clicking events button the user should go to events page which should show all the events happening in the city but closest events by distance should appear first then as distance increases events should be in that order. This page requires users location to show events so, if the user has provided a location already then it should not ask for location after coming on that page in the events page. If user has not provided any location then there should be a popup asking location to user.

My Prompt:

there should be an option to add location manually while providing location in the events page along with auto detech location option.

My Prompt:

the manual location option should not be lattitude and longitude it should be city or area or country and when user enters location in the field it should autosuggest location options eg. user types char it should suggest Charlotte, North Carolina, USA and Charleston, South Carolina, USA like this .

My Prompt:

Design and add to the current the Front Page for the Neighborrow App with Product Categories, Popular Listings, Testimonials, and Value Proposition Tiles Prompt Body: Create a modern, clean, and friendly landing page for a peer-to-peer item lending platform called Neighborrow. The design should reflect community, simplicity, and trust similar to peer-sharing platforms like Peerby or Nextdoor. Prioritize accessibility and mobile responsiveness. 🎯 Goals: Let users quickly browse product categories Promote popular listings in their local area (e.g., Boulder) Highlight community testimonials Communicate the value of sharing using benefit tiles Drive engagement with clear CTAs like “Join” or “Find out for yourself” ✅ Page Structure and Sections 1. Top: Product Categories Section ("All Products") Display horizontally scrollable tiles for major categories, each with an icon and label. Use a soft shadow and rounded cards for styling. Main Categories: Sound & Video Gaming Pets Tools Health, Wellness & Baby Clothing Household & Cleaning Event & Party Sports & Leisure Transportation Cooking Computers Subcategories under “Sound & Video”: Photography Photography Accessories Camcorders Other Audio & Video Music players Musical instruments Televisions Projectors & Accessories Music Accessories Audio equipment 2. Popular Listings (Location-Based Section) Create a section titled "Popular products", filtered by city (e.g., Boulder). Use a left-aligned vertical list of popular searches: Ipad rental in Boulder Headphones rental in Boulder Acoustic Guitar rental in Boulder Piano rental in Boulder Microphone rental in Boulder Microwave rental in Boulder Iron rental in Boulder At the bottom of the list, add a “More” link in purple. 3. Testimonials Section ("Why Members Love

Neighborrow") Create a 3-column card layout with user quotes, profile images, and locations. Examples: “I like Neighborrow for its ease of use...” Andy, Amsterdam “I found a jigsaw in 30 minutes...” Astrid, Utrecht “Helping people out by lending...” Breg, Amsterdam 4. Value Proposition Section ("It's good(s) for everyone") 4 white rounded cards with emojis/icons, titles in bold, and subtitles underneath. Examples: 🏡 Save & Earn On things you need only now and then 😀 Support your neighborhood Sharing feels good! 🌍 Do more with less And save raw materials, CO2, and waste 🛡️ Share Safely With guarantee against damage or loss Include a purple “Join” button centered beneath the tiles. 5. Frequently Shared Products Horizontally scrollable cards (image, title, subtitle, user avatar) under “Frequently Shared Products.” Example listings: Folding Table – “Fairly lightweight...” Jack Stand – “Free to borrow” Tent – “Great tent” Hand Truck – “Free to borrow” 🎨 Design Guidelines Use a neutral light background (#FFFFFF or #F8F8F8) Use Roboto or Inter font Primary Accent Color: #6A3DF5 (Neighborrow’s purple) Button color: Purple with white text Icon-based navigation where needed Shadowed cards with padding, white background, and subtle hover animations.

My Prompt:

Prompt for Generating Front-End Flow for NeighBorrow Web Application

Create a front-end implementation using React.js (with state management via React Hooks like useState and useContext) and React Router. The application flow includes:

Step-by-Step Requirements:

1. **Sign-Up Page:**
    - Form fields for user details (Name, Email, Password).
    - On submission, save data in local storage (simulate account creation, no backend needed).
    - After successful sign-up, redirect the user to the Login page with a success message: "Account created successfully. Please log in."
2. **Login Page:**
    - Login form (Email and Password).
    - Authenticate user details against the local storage entries.

   - On successful login, redirect the user to the Home Page with a welcome message.
3. **Home Page (Dashboard):**
   - Display local community events in card layouts (event name, date, location).
   - Include navigation buttons:
     - "Lend or Borrow Items"
     - Clicking navigates to an intermediate page to select action.
4. **Lend or Borrow Selection Page:**
   - Prompt: "Would you like to Lend or Borrow items?"
   - Two buttons: "Lend" and "Borrow".
     - "Lend" navigates to a lending page.
     - "Borrow" navigates to an items borrowing page.
5. **Lending Page:**
   - Form to list an item for lending (Item name, description, category, availability dates).
   - Save listings to local storage.
   - Display confirmation message upon listing: "Your item has been listed successfully."
6. **Borrowing Page:**
   - Display a list/grid of items available for borrowing from local storage.
   - Each item shows name, description, category, and availability.
   - Include a button to request borrowing; upon clicking, show a message: "Borrow request sent!"

Technical Stack Requirements:

- Front-end: React.js
- UI Styling: Tailwind CSS
- Routing: React Router
- Data Handling: Local Storage (front-end only, no backend integration)

Deliverables:

- Code snippets (JSX and JavaScript) for each component/page.
- Clear and concise documentation/comments explaining functionality and usage.
- Example local storage structures to demonstrate data storage and retrieval.

This implementation must be easily extendable for future integration with backend APIs.

My Prompt:

I'm building a community web app named **"NeighBorrow".** Currently, there are issues with the navigation header and the user authentication process. Here’s what I need your help with:

1. Header Navigation Issue:

- **Current Issue:** Navigation buttons ("Browse", "Share", "Events") are partially hidden underneath the header and become inaccessible when clicked.
- **Required Fix:** Ensure all header navigation buttons are clearly visible, fully accessible, and responsive on all device screens.

2. Authentication Persistence:

- **Current Issue:** After a user successfully signs in, clicking any navigation button logs them out unintentionally and redirects them to the home page.
- **Required Fix:** After signing in, users must remain authenticated until explicitly choosing to sign out via a dedicated "Sign Out" button.

3. Hamburger Menu Implementation:

- **Feature:** Once signed in, replace or complement header buttons with a responsive hamburger menu icon clearly visible.
- **Menu Items Include:**
    - **Profile:** Navigates to the user's Profile page.
    - **Lended Items (0):** Displays a count of items the user has lent out, initially "0" if none.
    - **Borrowed Items (0):** Displays a count of items borrowed by the user, initially "0" if none.
    - **Sign Out:** Explicitly logs the user out.

4. Profile Page Details:

- **Profile page** must show:
    - User’s **Name** and **Email.**
    - Option/button to **"Change Password".**
    - Option/button to **"Change Location".**

My Prompt:

I have two horizontal sections ("All Products" and "Frequently Shared Products") on my web application. Currently, they are not scrolling horizontally when I attempt to view more items using a mouse or trackpad. This makes additional items hidden and inaccessible to the user.

Detailed Requirements:

- Both sections ("All Products" and "Frequently Shared Products") should support **horizontal scrolling** using a mouse wheel or trackpad gesture.
- Ensure there's a clear indication (such as a scrollbar or arrows) that indicates additional items are available when scrolling.
- Maintain smooth scrolling behavior across all browsers and devices.
- Make sure the content does not overflow vertically or cause layout issues.
- Preserve existing styling, spacing, and responsiveness.

Sections to Fix:

1. **All Products**
   - Categories list horizontally displayed (Sound & Video, Gaming, Pets, etc.).
2. **Frequently Shared Products**
   - Items such as Folding Table, Jack Stand, Tent, etc., shown in horizontally scrollable cards.

Please provide clean and clear code changes with CSS adjustments or JavaScript snippets as needed. Ensure the provided solution integrates smoothly into my existing frontend built using React and Tailwind CSS.

My Prompt:

Thanks, it helped.
Now, I have a styling issue in my React-based web application (NeighBorrow App). On the lend/borrow page, when clicking on a dropdown selection (such as "Category"), the dropdown options

appear clearly, but the underlying content (such as headings and other text) is visible through the dropdown, causing overlapping and readability issues (see attached screenshot for reference). Requirements for the fix: Dropdown options should always appear above all other page elements. Ensure the dropdown menu background is solid and non-transparent, fully covering underlying text. Resolve any CSS-related stacking (z-index) or positioning issues. Maintain existing styles and responsive behavior. Please provide specific CSS or React/Tailwind CSS code modifications to resolve this dropdown overlap problem clearly and effectively.

My Prompt:

I have a React.js-based application called "NeighBorrow". Currently, I face two main usability issues:

1. Profile Page Enhancements:

- Currently, the Profile page shows only the option to "change location" but doesn't display the currently set user location.
- I need an additional field clearly displaying the user's current set location above or alongside the "Change Location" button.

Required Changes:

- Display the current user location clearly on the Profile page.
- Maintain existing functionality to change location.

2. Lending Management Enhancements:

- After lending an item, I want a dedicated page or dashboard view ("My Lended Items") to clearly show the list of items I have listed to lend.
- Under each listed item, I should clearly see incoming "Borrow Requests" from other users.
- For each borrow request, there must be clear and actionable buttons:
    - Accept Borrow Request
    - Decline Borrow Request

- Additionally, I must be able to view key profile details of users who sent the borrow requests, including:
  - User's name
  - User's current location

Required Functionalities:

- Create a "My Lended Items" page/dashboard to display all items that the logged-in user has listed.
- Implement UI to view borrow requests for each item, showing clearly the requesting user’s name and location.
- Include buttons to accept or decline each borrow request, triggering appropriate backend/API logic.

My Prompt:

I have a React.js and Tailwind CSS-based application named **"NeighBorrow".** Currently, the user experience around lending and borrowing items has some issues:

Issue 1: Lended Items Button Missing from Profile

- On the **Profile** page, add a button clearly labeled **"Lended Items".**
- Clicking this button should navigate users to a dedicated **"My Lended Items"** page.

Task:

- Add the **"Lended Items"** button prominently within the Profile page UI.

Issue 2: Manage Borrow Requests on Lended Items Page

- The new **"My Lended Items"** page must list all items the user has listed to lend.
- Under each lended item, clearly display any borrow requests received from other users.
- Each borrow request should include:
  - Requesting user's name
  - Requesting user's location
- Provide actionable buttons for each request:
  - **Accept Borrow Request**

    - **Decline Borrow Request**
- Implement appropriate frontend and backend logic upon clicking these buttons.

Task:

- Create a React component/page clearly listing lended items and incoming borrow requests.
- Integrate accept/decline functionality with backend API.

Issue 3: Prevent Users from Borrowing Their Own Listed Items

- Currently, users who list an item for lending can still see an option/button to borrow their own item.
- Modify the logic/UI so that if a user has listed an item:
    - They do **not** see the borrow option/button for that specific item.
    - Clearly indicate in the UI (if necessary) that the item is their own listing.

Task:

- Update conditional rendering logic in your React component to ensure users cannot borrow items they have personally listed.

Deliverables:

- Detailed, production-ready React component code snippets.
- Tailwind CSS usage for responsive and intuitive UI.
- API integration example snippets or explanations.
- Brief guidance on necessary backend updates or data schema modifications.

Make sure the UI changes follow best practices and maintain a smooth, intuitive user experience.

My Prompt:

Current Issue:

- After listing an item via the **lend page**, the item does **not appear** in the user's own **"Lended Items"** page.
- Other logged-in users cannot see the listed items in their neighborhood, hence can't request to borrow these items.
- The borrowing request and approval workflow is currently missing or incomplete.

Expected Functionality:
Implement the following clearly defined workflow:

1. Listing Items:

- When a user successfully lists an item on the **lend page**, this item should immediately appear on their personal **"Lended Items"** page.
- The item should also be visible to **other logged-in users** who are in the same area as the lender.

2. Borrowing Items:

- Other users viewing items in their area should clearly see the listed item and have a button or option to **"Request to Borrow"**.
- Clicking **"Request to Borrow"** sends a notification/request to the original lender.

3. Accept/Decline Borrow Requests:

- On the lender's **"Lended Items"** page, each listed item should clearly show incoming borrow requests with the requesting user's:
    - **Profile details** (Name and Location)
    - Buttons for **"Accept"** or **"Decline"** the borrow request.
- Once a request is **accepted**, the borrower should see a status update indicating they can pick up the item.

4. Borrowed Items Page:

- Add a new **"Borrowed Items"** page/tab accessible from the user's profile menu.
- On this page, borrowers see:
    - The status of their borrow requests (e.g., "Pending", "Accepted", "Declined").
    - Clear pickup instructions when a request is accepted.

Tasks for Cursor AI:

- Check current code (React.js with Tailwind CSS) to identify why listed items are not visible.
- Update state management (e.g., React state, Context API, or Redux) to track lent items and borrow requests accurately.
- Clearly implement and document new pages and components:
    - **"Lended Items" page** with accept/decline functionality.
    - **"Borrowed Items" page** showing request status.
- Ensure proper conditional rendering and notifications for user interactions.

Deliverables:

- Clearly commented code snippets for new React components/pages.
- State management improvements ensuring reliable data flow.
- Complete examples demonstrating borrow-request logic, request status updates, and notifications.
- Documentation of the implemented workflow to maintain clarity and future extensibility.

Ensure your solution follows React.js best practices and offers a seamless, intuitive user experience consistent with a community-sharing application.

My Prompt:

Current Issue:

I successfully listed an item from a user called **"YP"**, and it correctly appears in their **"Lended Items"** page. However, when I sign up and log in as a different user (**"ABCD"**) located in **Charlotte**, the item listed by "YP" does **not appear** on the borrow items page (`http://localhost:8080/borrow`). This borrow page is currently showing as blank (refer to the provided screenshot).

Expected Behavior:

- Any items listed by one user (**"YP"**) should be clearly visible and accessible on the borrow items page for **other users** (**"ABCD"**) logged in within the same area (Charlotte).
- Users should see a list/grid of items available to borrow, each clearly displaying:

    - Item name
    - Description
    - Location
    - Option to **"Request to Borrow".**

Tasks for Cursor AI:

1. Investigate the data-fetching logic on the `/borrow` page and determine why items listed by one user aren't visible to others.
2. Ensure the items listed in an area (like Charlotte) are correctly fetched and displayed to other users in the same geographic area.
3. Implement correct conditional rendering logic on the borrow page to clearly show items listed by different users.
4. Provide an actionable "Request to Borrow" button beside each listed item.

Technical Considerations:

- Frontend: React.js (with Tailwind CSS)
- Backend/Data Source: Check API calls fetching data related to borrowable items.
- State Management: Validate that fetched data populates correctly in React state/context.

Deliverables:

- Clearly documented and commented React component code for the borrow page.
- Updated API-fetching logic and data handling snippets demonstrating correct filtering and display based on location/user criteria.
- Ensure the solution follows React best practices for performance and user experience.

My Prompt:

I'm building a full-stack React.js + Node.js application called **Neighborrow.** I want to integrate it with **Supabase** as my backend database to handle authentication and data storage. Please help me:

✅ Objective:

Connect the Neighborrow app to Supabase and automatically set up:

1. **Users Table** (for auth and profile data)
2. **Lended Items Table** (items listed by users to lend)
3. **Borrowed Items Table** (borrow requests and approved items)
4. Generate all required files and folders for API logic and frontend integration.

🔧 Requirements:

1. Supabase Project Connection:

- Initialize Supabase in the project (`supabaseClient.js`).
- Use `.env` file for storing Supabase URL and API key securely.
- Create utility functions to handle Supabase queries (CRUD operations).

2. Tables to Be Created:

- **Users**
  - `id` (UUID, PK)
  - `email` (Text)
  - `name` (Text)
  - `location` (Text)
- **LendedItems**
  - `id` (UUID, PK)
  - `user_id` (FK to Users.id)
  - `title` (Text)
  - `description` (Text)
  - `category` (Text)
  - `location` (Text)
  - `is_available` (Boolean)
- **BorrowedItems**
  - `id` (UUID, PK)
  - `lended_item_id` (FK to LendedItems.id)
  - `borrower_id` (FK to Users.id)
  - `status` (Enum: 'pending', 'accepted', 'declined')
  - `requested_at` (timestamp)

3. File & Folder Structure (Auto-generate if missing):

- supabase/
  - `supabaseClient.js` (to initialize client)
  - `supabaseService.js` (for CRUD functions)
- api/
  - `lendedItems.js` (get, post, update)
  - `borrowRequests.js` (get, request, accept, decline)
- components/
  - LendItemForm.js
  - BorrowList.js
  - Profile.js
  - BorrowedItemsPage.js
- pages/
  - borrow.js
  - lend.js
  - lended-items.js
  - borrowed-items.js

⚙️ Functional Expectations:

- When a user signs up via Supabase Auth, they are auto-added to the Users table.
- When a user lends an item, it's stored in `LendedItems` with their `user_id`.
- When another user requests to borrow it, the request is logged in `BorrowedItems`.
- Borrowed and lended items should be fetchable and filterable by `user_id` and `location`.

🧪 Testing Instructions:

- Include a demo API call or form submission to create and fetch a lended item.
- Log successful creation and error handling via console.

📝 Deliverables:

- All necessary Supabase setup files
- Auto-generated schema and database tables
- CRUD functions for users, lended, and borrowed items
- Sample form/component integrations

- `.env` usage for secure keys

Make sure all code is clean, modular, and production-friendly.
Let me know if you'd like to create mock data for testing.