**TASK 1**  
**Product Name – Microsoft Surface Dial**  
(2016-Present)

**Product description:**

* The Surface Dial is a rotary input device designed for creative professionals.
* It is used to provide intuitive control over tools such as volume, zoom, brush size, and navigation in compatible apps.
* It was introduced as a DYNAMIC CONTROL INTERFACE, allowing users to access shortcuts, radial menus, media controls, and app-specific tools.

**Product features:**

* ROTARY TOUCH-SENSITIVE CONTROL: - Replaces traditional shortcut keys with a customizable rotating input dial.
* CUSTOMIZABLE SHORTCUTS: - Users could modify functions to quickly access commonly used commands.
* HAPTIC FEEDBACK: - Provides tactile response when rotating or pressing the dial for improved precision.
* BLUETOOTH CONNECTIVITY: - Enables wireless connection to Surface devices for a seamless experience.

**Product issues**:

* The Surface Dial relies on Bluetooth connectivity, which can sometimes cause latency, disconnection issues, or lag in response time.
* The inconsistent compatibility with non-Surface devices and limited support for third-party apps reduces its usability.
* Users reported that the dial’s sensitivity leads to accidental input, affecting workflow precision.
* The lack of advanced gestures makes it less versatile compared to other creative input devices.

**How to Improve (suggestions):**

* Instead of being a standalone device, Microsoft could introduce a HYBRID MODEL with both touch-sensitive controls and physical buttons for better accessibility.
* A more compact and customizable Surface Dial with multi-touch capabilities (e.g., pinch to zoom, swipe for navigation) could improve usability.
* Instead of being limited to Surface devices, a universal attachable version could be introduced, allowing users to attach it to any tablet, laptop, or desktop screen.
* Enhanced battery efficiency and an option to toggle between different sensitivity levels would optimize performance and reduce accidental inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | APPLICATION DOMAIN | COMPLEX PROBLEM IDENTIFIED | JUSTIFICATION |  |
|  |  |  |  |  |
| 1 | RIDE-SHARING APPLICATIONS (e.g., Uber, Lyft) | These platforms rely on **dynamic pricing algorithms**, which sometimes lead to **unexpected fare surges**. | Research has shown that **opaque pricing algorithms and surge pricing mechanics** may cause **dissatisfaction among users**, as fares fluctuate unpredictably, impacting affordability and trust. |  |
| 2 | VIDEO STREAMING APPLICATIONS (e.g., Netflix, Disney+) | Users have reported **video buffering and playback issues**, even with stable internet connections. | Multiple reports from user reviews highlight **content delivery network inefficiencies** and **poor adaptive streaming algorithms**, leading to inconsistent performance across devices. |  |
| 3 | MOBILE APPLICATIONS UTILIZING (e.g., Augmented Reality) | Integrating **AR technology into mobile apps** presents challenges related to **hardware limitations, real-time processing, and user accessibility**. | Studies have identified **high computational demands**, causing **increased battery drain and overheating**, along with **difficulties in rendering 3D models smoothly on all devices**. |  |

**TASK 2**

**TASK 3**

**CASE STUDY – Designing a Mobile App for Health Care**

**Problem - Difficulty in booking medical appointments efficiently.** **Limited access to medical records and health history.** **Unavailability of real-time health updates and emergency alerts**

**DESIGN THINKING PHASE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Objective** | **Application to the Problem** | **Solution Found** |
| **1. SCOPE** | Developing and testing a user-friendly mobile healthcare app. | With the growing focus on digital health, users are increasingly turning to mobile apps for tracking health metrics, booking doctor appointments, accessing medical records, and receiving virtual consultations. | The challenge is to develop and test a healthcare app that is intuitive, secure, and efficient, ensuring that users can easily track their health, schedule medical visits, receive real-time health updates, and consult with healthcare professionals. |
| **2. EMPATHIZE** | Understand the user's needs, challenges, and experiences. | Conduct surveys and interviews with patients, doctors, and caregivers to gather feedback about their pain points (difficulty scheduling appointments, accessing records, lack of telemedicine options, etc.). | Users want a seamless way to book appointments, access medical history, receive prescription reminders, and consult doctors virtually without long waiting times. |
| **3. DEFINE** | Define the problem clearly based on insights gathered. | Synthesize the data from the research phase. Identify key issues and user pain points. Create personas (e.g., elderly patients, working professionals, chronic disease patients). | "Users face challenges in managing their healthcare due to difficulties in booking appointments, lack of easy access to medical records, and limited remote consultation options. There is a need for a secure, easy-to-use platform that integrates all essential healthcare services." |
| **4. IDEATE** | Generate ideas and brainstorm potential solutions. | Brainstorm potential features: appointment scheduling, telemedicine, prescription tracking, symptom checker, AI-based health recommendations, emergency alerts. | Develop a feature list: virtual doctor consultations, electronic health record integration, AI-driven health insights, medication reminders, emergency contact features, real-time chat with healthcare providers. |
| **5. PROTOTYPE** | Build low-fidelity prototypes to visualize ideas. | Create wireframes or mockups of the app interface, focusing on key features like appointment booking, medical records access, and virtual consultations. | Develop clickable prototypes using tools like Figma or Sketch, showing how users can navigate the app to schedule appointments, consult doctors, and track prescriptions. |
| **6. TEST** | Test the prototype with real users to gather feedback. | Conduct usability testing with diverse users, including patients, doctors, and caregivers. Observe how they interact with the app and collect feedback. | Iteratively refine the app’s interface based on user feedback. For example, improving accessibility features for elderly users, integrating voice commands, or ensuring compliance with healthcare data privacy regulations. |
| **7. IMPLEMENT** | Develop the final product and launch it for wider use. | After refining the app based on user feedback, develop the final version and roll it out to a limited user base for real-world testing. | Full app launch with a seamless experience, secure data handling, and continuous user support. Collect data to assess long-term user satisfaction and future improvements. |

**TASK 4**

**1. Fixing a Product Configurator**

**Common Issues & Fixes**

| **Issue** | **Description** | **Possible Fixes** |
| --- | --- | --- |
| **Slow Performance** | **The configurator takes too long to load or respond.** | **Optimize queries, use caching, and reduce unnecessary computations.** |
| **Complex UI/UX** | **Users struggle to understand and navigate the tool.** | **Simplify UI, add tooltips, and introduce step-by-step guidance.** |
| **Incorrect Configurations** | **Users can select options that are incompatible.** | **Implement real-time validation and constraint-based logic.** |
| **Poor Mobile Experience** | **The configurator is not mobile-friendly.** | **Use responsive design and test across devices.** |
| **Lack of Integration** | **The tool does not sync with inventory, CRM, or ERP.** | **Use APIs to connect with external databases.** |
| **No Real-time Pricing Updates** | **Users don’t see updated prices as they configure products.** | **Implement dynamic pricing updates.** |
| **Limited Customization Options** | **Users want more flexibility in product selection.** | **Introduce AI-based recommendations for customization.** |

**2. CK Theory for Optimization**

**C-K Theory for Optimization**

**C-K (Concept-Knowledge) Theory is a framework for innovation and problem-solving that separates ideas into two domains: Concepts (C) and Knowledge (K). It enables structured exploration, iteration, and optimization by leveraging existing knowledge to develop new, improved solutions.**

**Step 1: Define the Problem (C0 – Initial Concept)**

**The optimization process begins with identifying inefficiencies or limitations in a system. This could be slow performance, high costs, low accuracy, or poor user experience.**

**Step 2: Analyze Existing Knowledge (K0 – Current Knowledge)**

**Gather and evaluate known information, including:**

* **Technical constraints (hardware, software, algorithms).**
* **Industry benchmarks (best practices, case studies).**
* **User feedback (pain points and desired improvements).**

**Step 3: Generate New Concepts (C1 – Idea Exploration)**

**By challenging existing assumptions, new concepts emerge. These may involve:**

* **Algorithmic improvements (AI-driven decision-making, automation).**
* **Structural changes (cloud integration, new frameworks).**
* **Enhanced user experience (personalization, intuitive UI).**

**Step 4: Test & Validate Concepts (K1 – New Knowledge Acquisition)**

**Concepts undergo experimentation, validation, and refinement through A/B testing, prototyping, and simulations. Some ideas are accepted into knowledge, while others are discarded.**

**Step 5: Implement & Iterate (C2 – Optimized Solution)**

**The best-performing concepts are integrated into the system, leading to a refined, optimized product. Continuous feedback loops allow further iterations, ensuring sustained improvements.**