Technische Umsetzungs-Evaluation:

Unreal Engine 5 als direkte Game Engine:
Unreal als Entwicklungsumgebung besitzt eine Masse an Tools, die den grundsätzlichen Implementationsaufwand drastisch reduzieren, diesen aber durch das Lernen der Umgebung eintauscht. Die Engine besitzt sowohl visuelles Scripting über sogenannte "Blueprints", was visuelles Entwickeln von Anwendungslogik ermöglicht. Außerdem besitzt die Engine einen sehr großen Asset, Texture Store und viele Beispielprojekte. Direkte Visualisierung von Daten mithilfe von Diagrammen oder Statistiken ist trotz vieler anderer nützlicher Tools nicht vorhanden. Auch liefert die Engine einen signifikanten Teil an Performance Overhead, weshalb sie im Android-Bereich normalerweise nur für AAA-Produktionen verwendet wird, die bereits hohe Performance-Requirements ansprechen und mit deren Nachteile weniger konfrontiert werden. Der Fokus liegt ganz klar auf Spiele- und Architekturprojekten, weshalb die sehr hohe Komplexität der Umgebung ein zusätzliches Risiko beim Einarbeiten kreieren könnte. Seit Mai 2025 gibt es ein offizielles Plugin “Data Charts” welches theoretisch viele der Visualisierungsprobleme durch direkte Diagramme innerhalb der Engine lösen könnte, dieses ist allerdings noch unbewehrt und liefert auch keine weitreichenden Anpassungen.

https://www.unrealengine.com/en-US/uses/mobile-games

https://www.unrealengine.com/en-US/feed?tags=interviews%2Cmobile

https://dev.epicgames.com/community/learning/tutorials/MPDr/fab-the-unreal-engine-data-charts-plugin

https://dev.epicgames.com/documentation/en-us/unreal-engine/android-support-for-unreal-engine

Web-basierte Anwendung mit JavaScript:
JavaScript bietet vielerlei Flexibilität im Umgang mit Daten über die populäre Library D3 welche sehr umfangreiche Tools bietet um die Visualisierung der Inhalte zu ermöglichen und anzupassen, besitzt allerdings auch eine steile Lernkurve welche in langer Sicht betrachtet auch ein Risiko darstellen könnte um schnell einen funktionierenden Prototypen zu entwickeln. Zusätzliche Flexibilität wäre durch die Browser-Umsetzung geboten, ohne die Anwendungslogik spezifisch für andere Plattformen stark anpassen zu müssen. Da JavaScript-Anwendungen jedoch häufig serverbasiert umgesetzt werden, würde dies bei einer rein lokal lauffähigen Anwendung unnötigen Mehraufwand erzeugen. Für das geplante Konzept könnte JavaScript allerdings auch vollständig clientseitig, also ohne Server- oder Internetabhängigkeit, umgesetzt werden.

https://d3js.org/

Kotlin Jetpack Compose und das Kotlin-Ökosystem:

JetBrains bietet mit Kotlins Jetpack Compose Library eine umfangreiche Entwicklungsumgebung im Android-Bereich für den User Interface Teil der App, mit dem das Team schon vertraut ist was somit maßgeblich die Ramp Up Zeit des Projekts beschleunigen würde. Zusätzlich ermöglichen die Tools Kotlin Notebooks mit Kandy und Lets-Plot die JetBrains zur Verfügung stellt ausreichende Hilfsmittel für das System-Konzept und dessen bildliche Darstellung ohne eine zu steile Lernkurve zu verlangen oder die Prototypisierung in die Länge zu ziehen. Die Produktivität könnte voraussichtlich durch den Mittelweg von Umfang und Komplexität, den Kotlin bietet, optimiert werden und bietet trotzdem für zukünftige Erweiterungsideen eine Grundlage, neue Tools einfach in das System zu integrieren.

https://kotlinlang.org/docs/data-analysis-visualization.html#create-a-bar-chart

https://kotlinlang.org/docs/lets-plot.html#before-you-start

https://developer.android.com/compose

Fazit:
Es wird daher empfohlen die technische Umsetzung mit Kotlin durchzuführen und es wird davon abgeraten Unreal Engine zu verwenden, als Fallback könnte JavaScript und D3 genutzt werden falls ungeahnte Komplikationen beim Start des Projekts auftreten oder die Anforderungen in einem Maße ändern sollten, die mit Kotlin nicht ausreichend erfüllbar sind.