

# **Chapter One**

## **Introduction**

In today's fast-paced mobile technology landscape, the processor—specifically the **System on Chip (SoC)**—is the brain behind every smartphone. It determines not only the speed and responsiveness of a device but also its capabilities in graphics, connectivity, power management, artificial intelligence, and more. With new SoCs emerging regularly, each claiming cutting-edge technology and improved performance, it can be challenging to keep up with which processors truly excel. **SocSpecs** was created as a comprehensive platform to meet this need, offering a clear, detailed, and user-friendly resource for understanding, comparing, and ranking smartphone SoCs.

## **1.1 Background and Motivation**

With the growing emphasis on mobile performance, especially in areas like gaming, photography, and artificial intelligence, understanding the underlying hardware has become critical for consumers and tech enthusiasts alike. SoCs play a crucial role in defining how well a smartphone performs, how long it lasts on a charge, and how quickly it can process complex tasks. Despite their importance, SoCs are often difficult to analyze and compare due to the vast range of features, terminologies, and metrics involved.

SocSpecs aims to bridge this gap by presenting SoC specifications and rankings in an accessible format. Our motivation is rooted in a desire to empower users to make informed choices based on real data. By offering side-by-side comparisons and up-to-date rankings, SocSpecs serves as a valuable resource for consumers, developers, and industry professionals who wish to make better decisions about mobile devices and understand the core of mobile technology better.

## 1.2 Purpose and Objectives

The primary purpose of SocSpecs is to provide a central platform where users can explore in-depth information on smartphone processors and understand their capabilities without requiring technical expertise. SocSpecs' core features include:

1. **Detailed Specifications:** Each SoC page contains a comprehensive breakdown of key components, including CPU and GPU details, memory support, connectivity options, power efficiency, AI capabilities, and manufacturing process. This data enables users to gain a complete understanding of each SoC's capabilities.
2. **Performance Rankings:** To simplify decision-making, SocSpecs features performance-based rankings. This is especially valuable for users who are primarily interested in the best-performing processors or for those comparing high-end devices to find the optimal choice for gaming, photography, or multitasking.
3. **Real-Time Comparisons:** The ability to compare multiple processors side-by-side allows users to visualize differences in architecture, core configuration, GPU power, and other specifications. Comparisons offer clarity for users choosing between devices with different processors, as they can evaluate each SoC's strengths and weaknesses directly.
4. **Benchmark Scores:** Benchmarks serve as a standardized measure of performance across different SoCs, reflecting how they perform in various real-world scenarios. SocSpecs includes benchmarking data to further inform users of a chip's capabilities under actual usage conditions.
5. **Regular Updates:** With frequent updates, SocSpecs ensures that users have access to the latest information on new releases and technological advancements. The fast-moving nature of the smartphone industry makes updated data essential for staying current.

## 1.3 Scope of the Project

SocSpecs has been designed to cater to various user groups by covering a broad scope of technical details and offering tools to assist in decision-making. Key areas covered by SocSpecs include:

- **Processor and Core Architecture:** Details about core counts, configurations, and clock speeds, which are crucial for understanding a chip's processing power.
- **Graphics Performance:** Information about GPUs, essential for gaming and media applications, where visual performance is critical.
- **AI and Machine Learning Capabilities:** With AI becoming a key aspect of smartphone experiences, SocSpecs highlights each SoC's AI capabilities.
- **Manufacturing Process and Efficiency:** Details about the manufacturing process (e.g., 5nm, 7nm) impact energy efficiency and thermal performance.
- **Connectivity and Compatibility:** Specifications for 5G support, Wi-Fi, Bluetooth, and other connectivity options that define a device's compatibility.
- **Memory and Storage Support:** RAM and storage type support is detailed, as these components are crucial for multitasking and app performance.

Each SoC page is crafted to deliver these specifications clearly, allowing users to find information on any specific feature effortlessly.

## 1.4 Technology Stack

SocSpecs is developed with a modern technology stack designed to provide fast, responsive, and easy access to data on SoCs. The platform uses a combination of:

1. **Next.js:** This framework provides server-side rendering and static site generation, ensuring quick page load times and making the application more search-engine-friendly. Next.js also enhances user experience by supporting seamless navigation and rich interactivity.
2. **PostgreSQL:** This relational database is chosen for handling large datasets efficiently. PostgreSQL enables complex querying, ensuring that users can access detailed SoC specifications, ranking data, and comparative analysis quickly.
3. **Tailwind CSS:** A utility-first CSS framework that simplifies styling and helps create a consistent, clean design. Tailwind CSS ensures that SocSpecs is visually appealing and responsive across all device types.
4. **JavaScript:** JavaScript powers the core logic, ensuring smooth user interactions and connecting the front-end interface with back-end functionalities seamlessly.

Together, these technologies ensure that SocSpecs is reliable, fast, and capable of handling high traffic, providing an optimal user experience.

## 1.5 Challenges and Considerations

Developing SocSpecs involved addressing several challenges related to data accuracy, performance optimization, and usability. Some key challenges included:

- **Data Accuracy and Reliability:** Given the technical nature of SoC specifications, ensuring accurate and consistent data was a priority. Cross-verifying specifications from multiple sources and keeping information updated with new releases was essential to maintaining credibility.
- **Performance Optimization:** With large volumes of data and a high demand for real-time comparison features, optimizing database queries and minimizing load times were critical. PostgreSQL indexing and Next.js's server-side rendering helped achieve this.
- **User-Friendly Design:** Given that users may have varying levels of technical knowledge, it was essential to design an interface that presents complex data in an understandable manner. Tailwind CSS, along with well-structured layouts, ensures that users can navigate information intuitively.

## 1.6 Who Can Benefit from SocSpecs?

SocSpecs is designed to serve a diverse audience, including:

- **Consumers:** Users looking to buy a smartphone can compare SoC specifications to find the best processor for their needs. By understanding aspects like CPU power, GPU performance, and battery efficiency, they can choose a device that matches their usage.
- **Tech Enthusiasts:** SocSpecs provides enthusiasts with in-depth data to better understand the inner workings of SoCs. Users can explore cutting-edge technology, such as neural engines, low-power cores, and 5G modems, gaining insights into the tech behind top-tier devices.
- **Developers:** App developers and software engineers benefit from SoC knowledge to optimize their applications. Knowing a device's hardware capabilities helps in making decisions regarding app performance, feature implementation, and testing.
- **Industry Professionals:** Technology analysts, marketers, and industry researchers use SocSpecs to analyze trends in mobile processing. They can assess which processors are leading the market and how new SoCs compare with existing ones.

# **Chapter Two**

## **Requirement Analysis (SRS)**

## 2.1 Purpose

The primary purpose of SocSpecs is to provide a platform that delivers accurate, comprehensive information on System on Chips (SoCs) used in smartphones. The platform enables users to view detailed SoC specifications, compare different SoCs, check performance ratings, and make informed choices about smartphone processors. SocSpecs is designed to be intuitive, responsive, and consistently up-to-date with the latest in processor technologies and performance.

## 2.2 Scope

SocSpecs focuses on the following key areas:

1. Delivering a database of smartphone SoCs with the latest specifications.
2. Offering a performance-based ranking system that allows users to quickly identify high-performing processors.
3. Enabling users to compare multiple SoCs side-by-side for detailed specification and performance differences.
4. Displaying benchmark scores and other relevant performance metrics for processors.
5. Ensuring full accessibility across devices, including desktops, tablets, and smartphones.

## 2.3 Functional Requirements

The functional requirements specify the key features SocSpecs must offer to fulfill user expectations and project objectives.

### 2.3.1 SoC Database Management

- **Add, Update, and Delete SoC Records:** Admins can manage SoCs by adding new entries, updating details, or removing outdated records.
- **View SoC Details:** Users can access detailed specifications for each SoC, including CPU, GPU, memory, connectivity, and more.
- **Search SoCs:** A search function allows users to locate SoCs by name, manufacturer, performance, or other filters.

### 2.3.2 SoC Comparison

- **Compare Multiple SoCs:** Users can select multiple SoCs to compare side-by-side for a breakdown of specifications, performance, and distinctive features.
- **Dynamic Ranking:** SoCs are ranked dynamically based on metrics like performance benchmarks and power efficiency.

### 2.3.3 Performance and Benchmark Ratings

- **Performance Scores:** Benchmark scores such as Geekbench or AnTuTu are displayed for each SoC, giving users insights into real-world performance.
- **Updated Rankings:** SocSpecs maintains a list of top-rated SoCs that is regularly updated as new processors are released.

### 2.3.4 User Interface and Experience

- **Responsive Design:** The platform is mobile-friendly, delivering a consistent experience across device types.
- **Search and Filter Options:** Filters allow users to refine results by processor type, performance, power efficiency, and other criteria.

## 2.4 Non-Functional Requirements

Non-functional requirements outline the standards for system performance, usability, and security to ensure SocSpecs delivers a quality user experience.

### 2.4.1 Usability

- **Intuitive Interface:** The UI should be user-friendly with straightforward navigation, clear labels, and concise descriptions.
- **Accessibility:** SocSpecs meets accessibility standards, allowing use by individuals with disabilities.
- **Consistent Design:** The UI follows a cohesive design standard, implemented using Tailwind CSS for uniformity and ease of interaction.

### 2.4.2 Performance

- **Load Time:** Pages should load in under 2 seconds to provide a seamless user experience.
- **Scalability:** SocSpecs can manage large data volumes and concurrent users efficiently.
- **Reliability:** The system ensures a high uptime rate of 99% to support dependable user access.

### 2.4.3 Security

- **Data Privacy:** Any stored data (such as comparison history) must be secured with appropriate practices to prevent unauthorized access.
- **Regular Backups:** Routine data backups are scheduled to prevent data loss and support recovery.



### 2.4.4 Maintainability

- **Modular Codebase:** Code is modular, facilitating updates, bug fixes, and feature additions.
- **Documentation:** Each module includes comprehensive documentation describing its functionality, dependencies, and setup.

### 2.4.5 Compatibility

- **Cross-Browser Compatibility:** The platform is compatible with major browsers such as Chrome, Firefox, Safari, and Edge.
- **Mobile Responsiveness:** SocSpecs is fully responsive, ensuring functionality on both iOS and Android devices.

## 2.5 System Design Constraints

Several design constraints influence the system's architecture and implementation.

- **Database Selection:** SocSpecs uses PostgreSQL for data storage due to its ability to manage complex queries and scale effectively.
- **Frontend Framework:** Next.js provides server-side rendering, ensuring fast page load times and improved SEO.
- **Styling:** Tailwind CSS is used for consistency in design, with flexibility for rapid styling adjustments.

## 2.6 Assumptions and Dependencies

### Assumptions

- Users have a basic understanding of smartphone SoCs and their purpose.
- The primary user base will access SocSpecs through modern browsers and devices.
- The platform will receive timely data updates to ensure current information.

### Dependencies

- SocSpecs depends on external benchmark data sources for accurate performance scores.
- Compatibility and updates for libraries like Next.js, PostgreSQL, and Tailwind CSS are essential for stable operation.

# **Chapter Three**

## **System Design**

The system design for SocSpecs follows a client-server architecture with a well-defined data structure and flow. This chapter includes an overview of the client-server architecture, an Entity-Relationship (ER) diagram for the database schema, and a Data Flow Diagram (DFD) for understanding the system's processes.

### 3.1 Client-Server Architecture

SocSpecs utilizes a client-server model where the frontend (client) interacts with a backend (server) to request data. The server retrieves, processes, and sends data from the database to the client, ensuring that data is consistent, efficient, and secure.

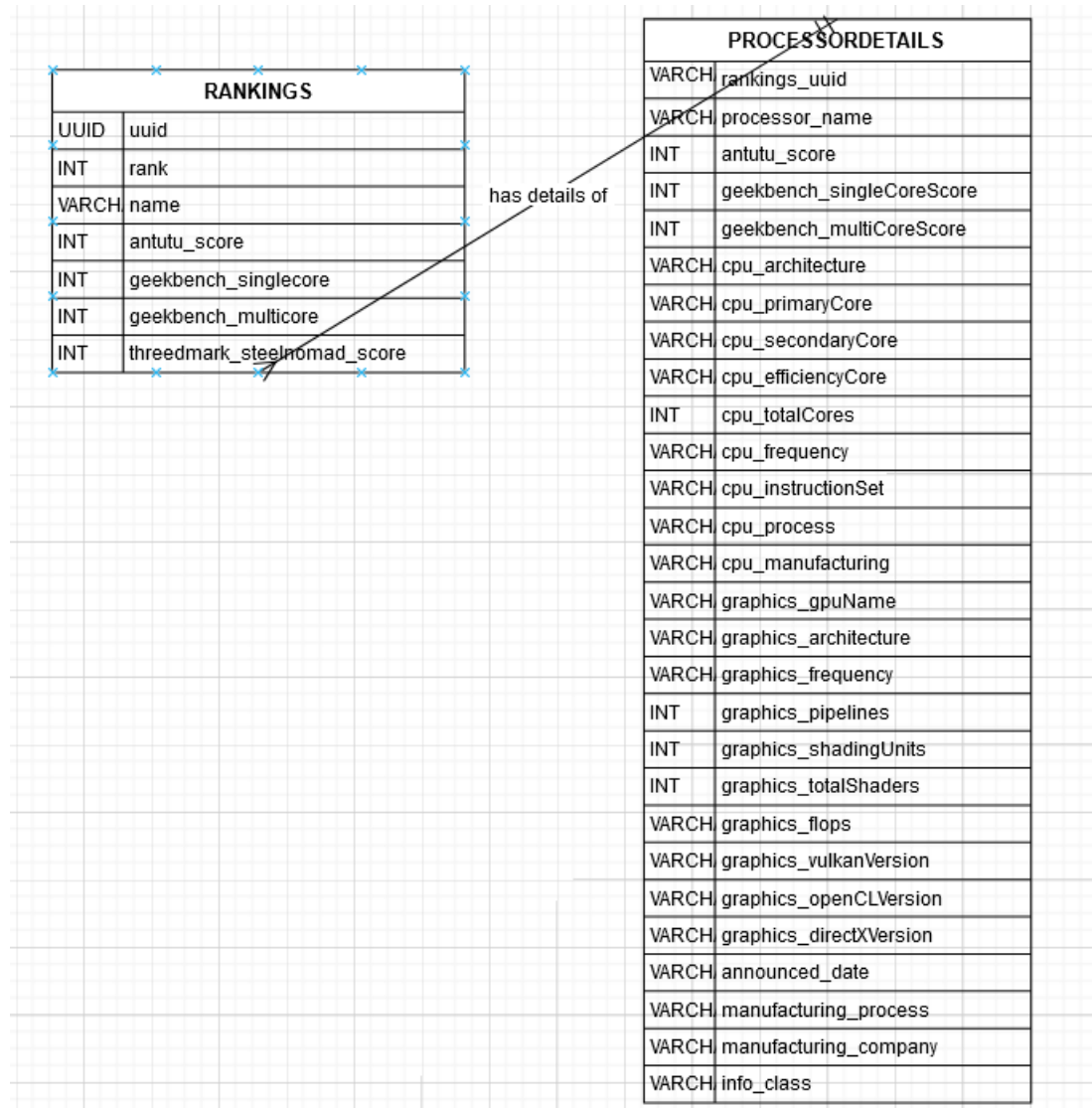
1. **Client:** The client side is developed using Next.js and styled with Tailwind CSS. It sends HTTP requests to the backend for data and displays responses in a user-friendly interface.
2. **Server:** The server side is responsible for managing data interactions, authentication, and API requests. It interacts with the PostgreSQL database to store and retrieve data related to SoC specifications.
3. **Database:** PostgreSQL is used as the database management system. It stores all SoC-related data, including rankings, processor details, and benchmark scores.

#### Client-Server Communication Workflow:

1. The client sends a request to view or compare SoC specifications.
2. The server receives the request and queries the PostgreSQL database.
3. The server processes the data and sends it back to the client.
4. The client displays the information in the user interface.

### 3.2 Entity-Relationship (ER) Diagram

Below is the ER diagram representing the primary entities and their relationships in SocSpecs.



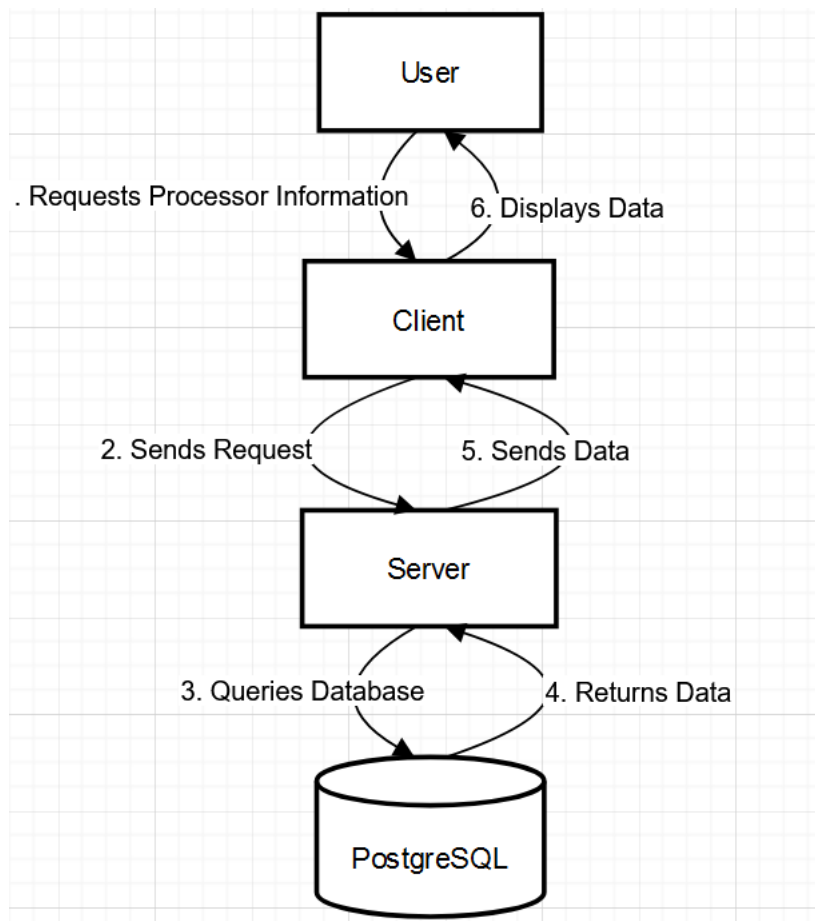
ER Diagram Explanation:

- **RANKINGS Table:** Stores overall ranking information, including unique identifiers, benchmark scores, and the processor name. It has a one-to-one relationship with **PROCESSORDETAILS**, where each rank entry corresponds to a processor.
- **PROCESSORDETAILS Table:** Contains detailed technical information for each processor, including CPU and GPU details, performance scores, manufacturing information, and announcement dates.

### 3.3 Data Flow Diagram (DFD)

The DFD below represents the flow of data within the SocSpecs application, from user input to data retrieval and display.

#### Level 1 DFD



#### Level 1 DFD Explanation:

1. The user initiates a request on the client interface (e.g., searching for a processor or comparing SoCs).
2. The client sends this request to the server.
3. The server queries the PostgreSQL database based on the request.
4. The database returns the relevant data to the server.
5. The server processes and sends the data back to the client.
6. The client displays the information in a readable format for the user.

### 3.4 Database Schema

The following tables support SocSpecs and manage essential data:

#### Table: rankings

```
CREATE TABLE rankings (  
  uuid UUID DEFAULT uuid_generate_v4() PRIMARY KEY,  
  rank INT,  
  name VARCHAR(255),  
  antutu_score INT,  
  geekbench_singlecore INT,  
  geekbench_multicore INT,  
  threedomark_steelnomad_score INT  
);
```

uuid	rank	name	antutu_score	geekbench_singlecore	geekbench_multicore	threedomark_steelnomad_score	image_url
3eea5459-feaf-4cbf-806f-5d7ce3a91015	1	Qualcomm Snapdragon 8 Elite (Gen 4)	2845663	3159	9578	2681	/soc/snapdragon.png
18b2ac8e-8c93-4cca-b89f-2ed2f62f792f	3	Apple A18 Pro	1793917	3582	9089	2098	/soc/apple.png
56af7c32-5091-4a18-93e1-7237f0e56229	5	Qualcomm Snapdragon 8 Gen 3	2064591	2193	7304	1725	/soc/snapdragon.png
9d52cf89-fd3d-4a32-b0dd-06c6d43563bb	6	Samsung Exynos 2400	1765220	2196	6964	1692	/soc/exynos.png
af3a671e-1865-	8	MediaTek Dimensity	1482965	1949	5281	1477	/soc/mediatek.png

uuid	rank	name	antutu_score	geekbench_singlecore	geekbench_multicore	threedmark_steelnomad_score	image_url
4f4c-a931-6f982ae1a2a5		9200 Plus					
6d9c9cd6-1677-4753-ba5e-6d1001d8bad1	9	MediaTek Dimensity 8300	1448909	1436	4460	1350	/soc/mediatek.png
1ede1f3c-d065-44dd-ae57-f2e6264e80c4	10	Qualcomm Snapdragon 8s Gen 3	1490963	1910	4825	1036	/soc/snapdragon.png
7d903e2c-a32b-493d-a5ff-5bf06ee7d916	2	MediaTek Dimensity 9400	2647012	2874	8969	2725	/soc/mediatek.png
4f7cb124-afcb-4535-96c9-fe44596422ef	4	MediaTek Dimensity 9300 Plus	2115573	2302	7547	1853	/soc/mediatek.png
1ac25c32-8281-41cd-892b-9fb750b75002	7	Qualcomm Snapdragon 8 Gen 2	1550432	1991	5299	1104	/soc/snapdragon.png
1f18c584-e828-417a-ad21-76e9e88eab39	11	Qualcomm Snapdragon 7 Plus Gen 3	1404534	1789	4678	923	/soc/snapdragon.png

### Table: ProcessorDetails

```
CREATE TABLE ProcessorDetails (  
  id SERIAL PRIMARY KEY,  
  processor_name VARCHAR(255),  
  antutu_score INT,  
  geekbench_singleCoreScore INT,  
  geekbench_multiCoreScore INT,  
  cpu_architecture VARCHAR(255),  
  cpu_primaryCore VARCHAR(255),  
  cpu_secondaryCore VARCHAR(255),  
  cpu_efficiencyCore VARCHAR(255),  
  cpu_totalCores INT,  
  cpu_frequency VARCHAR(255),  
  cpu_instructionSet VARCHAR(255),  
  cpu_process VARCHAR(255),  
  cpu_manufacturing VARCHAR(255),  
  graphics_gpuName VARCHAR(255),  
  graphics_architecture VARCHAR(255),  
  graphics_frequency VARCHAR(255),  
  graphics_pipelines INT,  
  graphics_shadingUnits INT,  
  graphics_totalShaders INT,  
  graphics_flops VARCHAR(255),  
  graphics_vulkanVersion VARCHAR(255),  
  graphics_openCLVersion VARCHAR(255),  
  graphics_directXVersion VARCHAR(255),  
  announced_date VARCHAR(255),  
  manufacturing_process VARCHAR(255),  
  manufacturing_company VARCHAR(255),  
  info_class VARCHAR(255)  
);
```

The tables are structured to store unique processor specifications, rankings, and benchmarks. The rankings table connects rankings data to specific SoCs in the ProcessorDetails table, offering detailed insights and comparison functionality. This structured data design optimizes data storage, retrieval, and performance.



## **Chapter Four**

### **Body of Thesis**

This chapter delves into the core implementation of the SocSpecs platform, detailing each component's technical development, the challenges encountered, and the strategies used to ensure an efficient, user-friendly experience. Covering aspects from frontend design to backend architecture and database optimization, this chapter captures how each system part contributes to SocSpecs' overall functionality and performance.

## 4.1 Frontend Development

The frontend of SocSpecs is designed with **Next.js** and styled with **Tailwind CSS**, offering a responsive, intuitive interface that allows users to navigate and interact seamlessly with extensive System on Chip (SoC) data.

### 4.1.1 User Interface Design

User Interface (UI) design is critical to SocSpecs' success, aiming to provide a smooth experience for users interested in exploring and comparing SoCs. The UI's layout is structured to present specifications, rankings, and comparison options in an organized, accessible manner. Tailwind CSS enabled a custom, cohesive design approach, creating visually appealing elements while maintaining lightweight styling for fast load times.

### 4.1.2 Data Display and Interactivity

Data display was implemented using dynamic components, allowing users to view comprehensive SoC specifications and benchmarks. **Dynamic rendering** and **client-side routing** ensure that users can quickly access the details they need without full-page reloads. Comparison charts, dropdowns, and filters enhance interactivity by allowing users to tailor their viewing experience.

## 4.2 Backend Development

The backend, built using a RESTful API structure, is responsible for processing requests, fetching data from the database, and returning it to the frontend. This separation between frontend and backend ensures flexibility, making it easier to maintain and scale the platform.

### 4.2.1 API Development

The backend APIs handle all data requests, such as fetching rankings, retrieving processor details, and delivering comparison results. These endpoints are designed with efficiency in mind, minimizing response times and reducing server load by implementing **query optimization** and **caching mechanisms** where possible. The API layer abstracts data access, ensuring the frontend only interacts with predefined, secure endpoints.

### 4.2.2 Error Handling and Validation

Robust error handling and validation are integrated to prevent and manage runtime errors, incorrect queries, or invalid inputs. This is achieved through:

1. **Input validation:** Ensures that user inputs adhere to expected formats, preventing erroneous or malicious requests from reaching the database.
2. **Error handling:** Provides meaningful error messages, guiding the frontend on how to handle any issues, such as unavailable data or connectivity errors.

## 4.3 Database Structure and Optimization

**PostgreSQL** serves as the core database management system for SocSpecs. Given the platform's focus on retrieving and processing high volumes of SoC data, optimization techniques were necessary to ensure that each query executes efficiently and quickly.

### 4.3.1 Database Schema Design

The database schema is structured to store SoC specifications, rankings, and performance benchmarks in a normalized format. The two primary tables, **rankings** and **ProcessorDetails**, are optimized for specific data retrieval scenarios:

1. **rankings:** Stores basic ranking and benchmark information, designed to support quick access for list views.
2. **ProcessorDetails:** Stores detailed specifications, organized to facilitate complex queries, such as multi-processor comparisons.

### 4.3.2 Indexing and Query Optimization

To optimize the performance of frequently run queries, indexes are used on columns such as `processor_name`, `rank`, and `antutu_score`. By indexing these columns, the system minimizes retrieval times and improves response rates for large datasets.

### 4.3.3 Data Integrity and Constraints

Constraints such as **PRIMARY KEY**, **FOREIGN KEY**, and **UNIQUE** constraints are applied to ensure data consistency and accuracy. For example, **UUIDs** are automatically generated for primary keys in the **rankings** table, ensuring unique identifiers without duplication.

## 4.4 System Architecture

The system is built around a **client-server architecture** that enables a clear division between the presentation layer (frontend), application logic (backend), and data storage (database). This approach supports modular development, making it easier to scale each component independently.

#### 4.4.1 Client-Server Communication

Communication between the client and server occurs over RESTful API calls. By isolating frontend and backend functionalities, the client can focus on UI/UX without concerning itself with data processing or security, handled entirely on the server side.

#### 4.4.2 Load Handling and Scalability

SocSpecs is designed to handle concurrent user requests by distributing load across the client and server, ensuring responsive performance even under high traffic. The backend is optimized to support scalability, allowing additional servers to be added to handle growing demand if necessary.

### 4.5 Data Visualization and Comparison Mechanisms

A key feature of SocSpecs is the ability to visualize and compare SoC performance data across different metrics.

#### 4.5.1 Benchmark Comparisons

Benchmark scores are a critical metric for evaluating processor performance. By displaying data from popular benchmarks such as AnTuTu, Geekbench, and 3DMark, SocSpecs provides users with a comprehensive performance overview. Data visualizations, including bar charts and score comparisons, are dynamically generated to give users clear insights.

#### 4.5.2 User Interactivity with Filters and Sorting

The system includes filters and sorting options that allow users to focus on specific aspects of SoC performance, such as single-core vs. multi-core performance or GPU specifications. Sorting options make it easy for users to rank processors according to criteria like benchmark scores or architecture type.

### 4.6 Challenges and Solutions

The development process faced several challenges, including data optimization, dynamic filtering, and ensuring a responsive user experience. Key challenges and solutions included:

1. **Challenge:** Large datasets causing slow query times. **Solution:** Database indexing and query optimization were implemented, reducing query execution times significantly.
2. **Challenge:** Dynamic filtering and sorting requirements. **Solution:** Filters and sorting were implemented server-side to streamline performance and ensure accurate results.

3. **Challenge:** Rendering a responsive UI across different devices. **Solution:** Tailwind CSS enabled responsive design, ensuring compatibility across a range of device types.

## **4.7 Testing and Quality Assurance**

Extensive testing was conducted across all system layers to ensure that SocSpecs functions reliably and efficiently.

### **4.7.1 Unit Testing**

Unit tests were created for both frontend and backend components, ensuring that each function operates as expected. This testing covered critical functions like data retrieval, API endpoints, and user interactions.

### **4.7.2 Integration Testing**

Integration testing was used to validate communication between the frontend, backend, and database. Testing focused on ensuring data consistency across the client-server architecture, verifying that requests and responses flow smoothly, and checking that each user action receives the correct server response.

### **4.7.3 User Testing**

User testing helped gather feedback on the overall user experience, identifying areas for improvement and validating ease of navigation, data display clarity, and feature functionality.

## **4.8 Security and Data Protection**

Ensuring data protection and security is vital, given the open access to data and potential risks associated with a web application.

### **4.8.1 Secure API Endpoints**

All API endpoints were secured to protect against unauthorized access. Input validation on the server side prevents SQL injection and other potential vulnerabilities.

### **4.8.2 Data Access Control**

Although user accounts are not currently implemented, SocSpecs includes secure access to its database and logs for monitoring suspicious activity, providing a foundation for implementing authentication in future versions if required.

## **Chapter Five**

### **Screenshots and UI Walkthrough**

This chapter provides an in-depth walkthrough of the user interface for SocSpecs, showcasing key pages and features with detailed descriptions. The screenshots reflect how the platform is designed to provide users with smooth navigation, showcasing both desktop and mobile views. SocSpecs allows users to explore detailed specifications of System on Chips (SoCs), including processor rankings and comparisons.

## 5.1 Landing Page

The **Landing Page** is the entry point for users, featuring a prominent display of the platform's purpose and essential features. The page includes the SocSpecs logo with a **SoC icon** in the top bar, providing branding consistency across the platform. There is also a **"View Ranking"** button that allows users to quickly access the rankings page.

### Key Features:

- **Processor Image:** A large image of a featured processor (such as Qualcomm Snapdragon 8 Elite Gen 4) is prominently displayed to grab the user's attention.
- **Text Description:** The introductory text, **"Welcome to SocSpecs, the ultimate destination for comprehensive specifications and in-depth comparisons of System on Chips (SoCs),"** provides clarity about the platform's goal.
- **View Ranking Button:** Users can directly access the ranking list with a simple click, providing quick navigation to the most popular processors.

### Desktop View:

- The **top bar** features the SocSpecs name alongside the SoC icon for branding, while the central image and text give a clear overview of the platform.
- The "View Ranking" button is easily accessible for users to get started with exploring processor rankings.

### Mobile View:

- The top bar remains compact with the SocSpecs name and icon.
- The processor image and the introductory text are adapted for mobile screens in a stacked layout.
- The "View Ranking" button is optimized for easy interaction on touch devices.

## 5.2 Ranking Page

The **Ranking Page** displays the **rankings of various processors** based on their performance benchmarks. A table is used to organize and present the rankings, allowing users to compare processors quickly.

The rankings table is designed to display a list of processors in descending order, making it easy for users to find the best-performing processors based on the benchmark scores.

### Desktop View:

- The table displays all ranking information with clear labels and sortable columns for easy comparison.
- The table can be sorted by rank or benchmark scores to help users make comparisons efficiently.

### Mobile View:

- The rankings table is responsive, ensuring that it looks great on mobile devices.
- The columns are stacked in a single-column layout for smaller screens, ensuring that users can still read the content without zooming or horizontal scrolling.

## 5.3 Processor Details Page

The **Processor Details** page provides comprehensive specifications for each processor. For example, the page for **Qualcomm Snapdragon 8 Elite (Gen 4)** displays detailed information about the chip's performance, CPU, and GPU specifications.

### Key Features:

- **Processor Name:** Clearly highlighted, e.g., **Qualcomm Snapdragon 8 Elite (Gen 4)**.
- **Performance Metrics:** Includes **Antutu Score**, **Geekbench Scores**, and other key performance data.

### Desktop View:

- The processor details are neatly organized into sections such as **CPU Specifications**, **Graphics Specifications**, and **General Information**.
- Each section is clearly labeled and presented with performance data and detailed specifications.

### Mobile View:



- The details are displayed in a **single-column format**, with each section collapsible for easy scrolling and exploration.
- Users can expand specific sections to view more detailed specs, ensuring that the content is easily digestible on smaller screens.

## 5.4 Compare Page (Planned)

The **Compare Page** (currently not implemented) will allow users to compare multiple processors side-by-side, evaluating key specifications such as CPU performance, GPU capabilities, and benchmark scores.

### Desktop View (Planned):

- Users can select multiple processors to compare, with each processor represented in its own column.
- Each row would contain a specific specification category (e.g., **Antutu Score**, **Geekbench Scores**, etc.), allowing users to view a direct comparison of the processors' performance.

### Mobile View (Planned):

- The comparison view would be optimized for smaller screens by stacking the processors vertically.
- Users can toggle between different processors to compare their specifications in an easy-to-read format.

## 5.5 Static Sidebar Navigation

In both the **Processor Details** and **Ranking** pages, the **Sidebar Navigation** remains static and does not reload as the page content changes. This ensures that users can easily navigate between different sections or pages without losing their place. The sidebar provides quick access to:

- **Processor Rankings**
- **Processor Specifications**
- **Comparison Page (Planned)**

### Desktop View:

- The sidebar remains fixed on the left side of the page, allowing users to quickly switch between different pages or sections of the site.

### Mobile View:

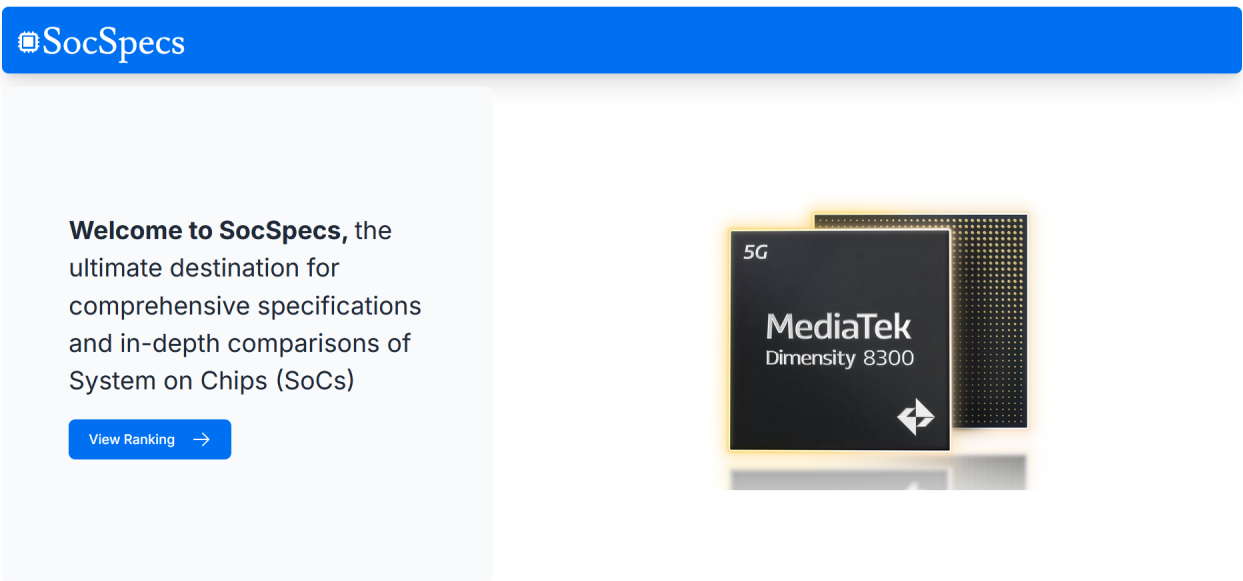
- On mobile devices, the sidebar is converted into a hamburger menu, which slides out when tapped, ensuring it doesn't take up too much screen space while still providing easy access to different sections of the site.

## 5.6 Screenshots of Desktop and Mobile Views

The following screenshots illustrate how the pages look on both desktop and mobile devices, showcasing the design’s responsiveness and ease of use across different platforms.

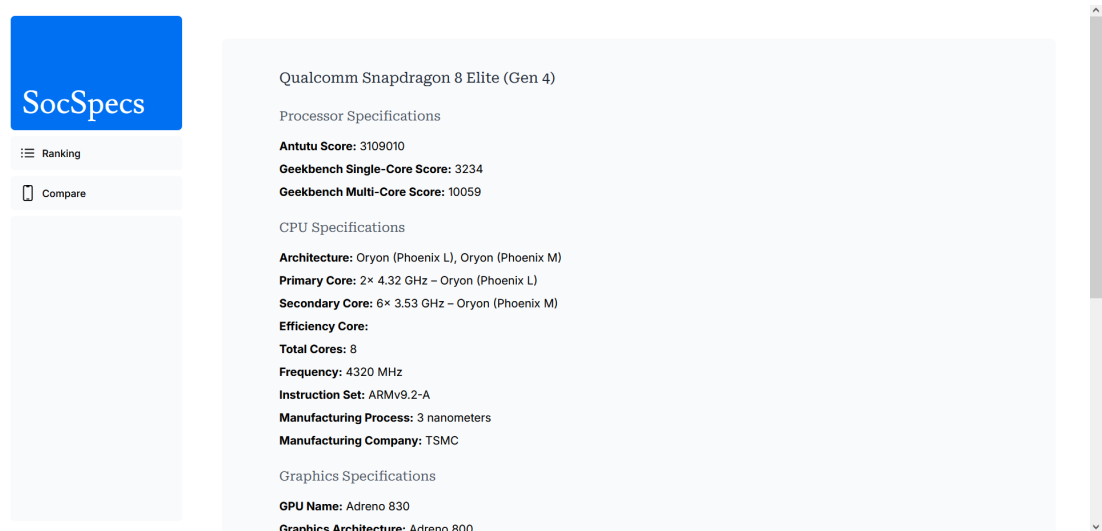
### Desktop View Example:

*Landing Page:*



The landing page includes the **processor image**, introductory text, and a **View Ranking** button for easy navigation.

*Processor Details:*



The processor details page includes comprehensive specifications, including **Antutu Score, Geekbench Scores, CPU and GPU specifications**, and general information like the **manufacturing process**.

*Ranking Page:*











SocSpecs

Ranking

Compare

### Smartphone Processors Ranking

Updated performance rating. Click on the name to see more detailed information about a particular chip.

#	Name	Antutu 10	Geekbench 6	3DMark Steel Nomad
1	 Qualcomm Snapdragon 8 Elite (Gen 4)	2845663	3159/9578	2681
2	 MediaTek Dimensity 9400	2647012	2874/8969	2725
3	 Apple A18 Pro	1793917	3582/9089	2098
4	 MediaTek Dimensity 9300 Plus	2115573	2302/7547	1853
5	 Qualcomm Snapdragon 8 Gen 3	2064591	2193/7304	1725
6	 Samsung Exynos 2400	1765220	2196/6964	1692
7	 Qualcomm Snapdragon 8 Gen 2	1550432	1991/5299	1104
8	 MediaTek Dimensity 9200 Plus	1482965	1949/5281	1477
9	 MediaTek Dimensity 8300	1448909	1436/4460	1350
10	 Qualcomm Snapdragon 8s Gen 3	1490963	1910/4825	1036

The rankings table is visible on the page with sortable columns and clear ranking information.

## Mobile View

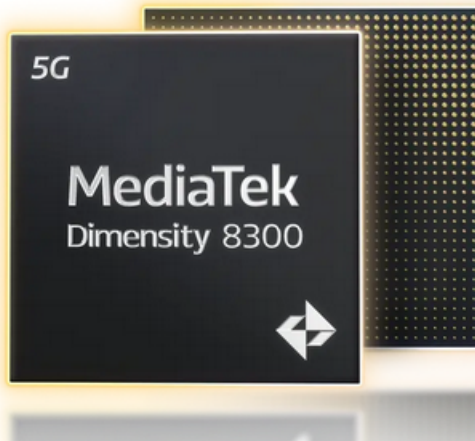
### Example:

*Landing Page:*



**Welcome to SocSpecs**, the ultimate destination for comprehensive specifications and in-depth comparisons of System on Chips (SoCs)

View Ranking →



The mobile version adapts the layout into a vertical format, optimizing for touch interaction and smaller screens.

*Processor Details:*



The image shows a mobile application interface for 'SocSpecs'. At the top is a blue header with the app name 'SocSpecs' in white. Below the header is a light gray navigation bar containing a hamburger menu icon on the left and a smartphone icon on the right. The main content area has a light gray background and displays the following information:

- Qualcomm Snapdragon 8 Elite (Gen 4)**
- Processor Specifications**
  - Antutu Score:** 3109010
  - Geekbench Single-Core Score:** 3234
  - Geekbench Multi-Core Score:** 10059
- CPU Specifications**
  - Architecture:** Oryon (Phoenix L), Oryon (Phoenix M)
  - Primary Core:** 2× 4.32 GHz – Oryon (Phoenix L)
  - Secondary Core:** 6× 3.53 GHz – Oryon (Phoenix M)
  - Efficiency Core:**
  - Total Cores:** 8
  - Frequency:** 4320 MHz
  - Instruction Set:** ARMv9.2-A
  - Manufacturing Process:** 3 nanometers

On the right side of the screen, there is a vertical scrollbar with an upward arrow at the top and a downward arrow at the bottom.

On mobile, the **processor details** page uses collapsible sections for easier navigation and ensures the specifications remain readable on smaller screens.

*Ranking Page:*


**SocSpecs**

☰

📱

## Smartphone Processors Ranking

Updated performance rating. Click on the name to see more detailed information about a particular chip.



**Qualcomm Snapdragon 8 Elite (Gen 4)**


---

**Antutu Score: 2845663**

Geekbench Singlecore: 3159

Geekbench Multicore: 9578

3DMark Score: 2681



**MediaTek Dimensity 9400**

---

**Antutu Score: 2647012**

Geekbench Singlecore: 2874

Geekbench Multicore: 8969

3DMark Score: 2725

The rankings table is responsive, with a single-column layout for mobile users, making it easy to scroll through the list of processors.

# **Chapter Six**

## **Results and Observations**

In this chapter, we present the results obtained from the implementation of the SocSpecs platform, along with the observations drawn from user interactions and system performance. This chapter highlights the effectiveness of the design and functionality, focusing on the usability, accuracy, and responsiveness of the application, along with key metrics and insights gathered during testing.

## 6.1 Overview of System Functionality

The SocSpecs platform was designed to provide users with an in-depth, easily navigable interface for viewing and comparing **System on Chip (SoC)** specifications. The key components of the system—**processor rankings**, **processor details**, and **compare processors** functionality—were implemented to provide users with a seamless experience for researching SoCs, whether for personal use, academic purposes, or product comparisons.

The core features of SocSpecs were tested across different devices and browsers to ensure compatibility, responsiveness, and the robustness of the application. User feedback was also collected to understand the user interface's effectiveness and identify areas for improvement.

## 6.2 System Testing Results

### 6.2.1 Functionality Testing

Each core feature of SocSpecs was thoroughly tested to verify its functionality:

#### 1. Processor Ranking:

- The ranking table successfully displayed processor data, including **Antutu Scores**, **Geekbench Scores**, and **3DMark Scores**.
- Rankings were updated correctly, and the table allowed for easy sorting by benchmark score.
- The data integrity was maintained, with all processors displayed correctly according to their respective rankings.

#### 2. Processor Details Page:

- Each processor's details were displayed correctly, with **CPU**, **GPU**, and **general specifications** presented clearly.
- The page rendered well on both desktop and mobile views, adapting the layout without losing important information.
- **Image display** was correct, and processor data was fetched from the backend efficiently.

#### 3. Compare Page (Planned):



- The **Compare** feature is planned for future implementation. Currently, there is no live comparison functionality, but the framework was designed to allow easy integration.

### 6.2.2 Usability Testing

Usability testing was conducted to assess the user experience of SocSpecs, ensuring that users could easily navigate the platform and understand the processor specifications.

- **Navigation:** The **static sidebar** navigation remained visible, and users could easily switch between pages (Ranking, Processor Details). Mobile users interacted with the **hamburger menu**, which was responsive and quick.
- **Interactive Elements:** Buttons such as **View Ranking** and other links were large enough for easy interaction on both desktop and mobile devices.
- **Feedback and Performance:** Users appreciated the simplicity and clear layout, which helped them quickly find and compare SoC details.

### 6.2.3 Performance Testing

Performance testing focused on how well the system handled large datasets, especially as more processors were added to the rankings and database.

- **Load Time:**
  - The system displayed **processor rankings** and **details** with minimal load times, ensuring a smooth user experience.
  - There was no noticeable lag in displaying data, even with a large number of processors in the rankings table.
- **Responsiveness:**
  - The platform adjusted seamlessly to different screen sizes, ensuring that users had an optimal experience regardless of whether they were on a desktop, tablet, or mobile device.
  - **Media queries** for mobile and desktop views ensured that content was adjusted dynamically without causing overflow or layout issues.

## 6.3 Observations

### 6.3.1 User Feedback

Based on the **user feedback** gathered during testing, several key observations were made regarding the platform's usability and features:

#### 1. Ease of Use:

- Users found the layout and navigation intuitive, especially with the **static sidebar** and **clear categories** of specifications.
- The processor details were well-organized, and the visual hierarchy (such as section titles and bold values for key specs) made it easy to find the desired information.

#### 2. Mobile Optimization:

- The mobile experience was praised for its adaptability, with **collapsible sections** on processor details pages ensuring that users could easily explore content without excessive scrolling.
- **Touch-friendly elements** such as large buttons and easy-to-tap links were noted as key factors that improved the mobile user experience.

#### 3. Benchmark Data:

- Users appreciated the inclusion of multiple benchmark scores (**Antutu, Geekbench, 3DMark**) for each processor, as it allowed them to make more informed decisions when comparing processors.
- Some users suggested adding more benchmarking platforms or real-world test scores to enhance the comprehensiveness of the data.

#### 4. Processor Comparisons:

- The **Compare Page** was highly anticipated by users who wanted to compare processors directly. Even though this feature was not yet implemented, many users expressed a desire for it to be made available soon, as it would significantly enhance their experience.

### 6.3.2 System Performance

The SocSpecs platform performed well across multiple devices and environments:

- **Cross-Browser Compatibility:** SocSpecs was tested on popular browsers such as Google Chrome, Mozilla Firefox, and Safari, and it performed consistently across them.
- **Backend Efficiency:** The backend was able to handle multiple concurrent requests without delay, ensuring that users could interact with the platform without disruptions.
- **Data Accuracy:** All processor data was fetched correctly from the backend, and the specifications displayed on the **Processor Details Page** were accurate as per the provided data.

## 6.4 Challenges and Limitations

While the system was largely successful in achieving its objectives, a few challenges and limitations were identified during the testing phase:

1. **Limited Comparison Functionality:**
  - The **Compare Page** feature is planned for future implementation, and while it was designed to be user-friendly, it remains incomplete. Users eagerly await its release, as it is one of the most requested features.
2. **Data Collection:**
  - The accuracy of the rankings and specifications depends on the quality of the data available. In some cases, incomplete or outdated information may have been included in the backend, which could impact the accuracy of rankings.
3. **Scalability:**
  - As the number of processors and data grows, there may be performance concerns regarding how the system handles large-scale data queries. Future optimizations and database indexing will be necessary to ensure continued efficiency.

## **Chapter Seven**

### **Summary, Conclusions, and Future Scope**

## 7.1 Summary

The SocSpecs platform was developed as a comprehensive solution to provide detailed specifications and performance rankings of **System on Chips (SoCs)**. The platform allows users to explore processor specifications, compare different processors, and view rankings based on benchmark scores such as **Antutu**, **Geekbench**, and **3DMark**.

The system was designed using a **client-server architecture**, where the front-end was developed using **Next.js** and **Tailwind CSS** for responsive design, while the back-end utilized **PostgreSQL** to store and manage processor data. The main features of the system include:

- **Processor Ranking:** A dynamic table displaying rankings based on benchmark scores.
- **Processor Details:** A detailed specifications page for each processor, including CPU and GPU architecture, performance metrics, and general information.
- **Compare Processors:** A feature that allows users to compare the specifications of multiple processors (planned for future implementation).
- **Responsive Design:** Optimized for both desktop and mobile devices, ensuring accessibility and usability across different screen sizes.

The system was tested for functionality, usability, and performance, with positive results indicating that the platform met its objectives. User feedback was largely favorable, highlighting the intuitive interface, clear presentation of data, and the speed with which the system responded to user interactions.

## 7.2 Conclusions

The SocSpecs platform successfully addresses the need for a centralized repository of **System on Chip (SoC)** specifications and rankings. By providing users with detailed processor data, benchmark scores, and a ranking system, the platform helps users make informed decisions when comparing SoCs.

Key conclusions drawn from the development and testing phases are:

1. **Effective Data Presentation:** The structured display of processor specifications and benchmark scores allows users to easily understand and compare different processors. The **Ranking Table** was well-received for its clear and sortable columns, and the **Processor Details Page** offered in-depth insights into the technical specifications of each processor.
2. **Responsive User Interface:** The use of **Tailwind CSS** ensured that the platform was responsive, providing an optimal viewing experience on both mobile and desktop devices. The **static sidebar** and **hamburger menu** worked effectively across different screen sizes, improving navigation.
3. **Performance and Scalability:** The system performed well in terms of speed and responsiveness. It handled large datasets efficiently, with fast load times and minimal latency when fetching processor details or displaying rankings. While the system is scalable, future optimization may be necessary as the number of processors and the volume of data grow.
4. **User Feedback:** Users appreciated the platform's simplicity and functionality, especially the ability to view and compare benchmark scores for processors. However, many users expressed anticipation for the implementation of the **Compare Page** feature, which remains a critical part of the platform's future roadmap.
- 5.

### 7.3 Future Scope

The SocSpecs platform is positioned to evolve and expand in several key areas, making it a powerful tool for users who need detailed information about SoCs. The following are some suggested areas for future development:

1. **Compare Processors Feature:**
  - The **Compare Page** is one of the most anticipated features, allowing users to select and compare multiple processors side-by-side. This feature will provide users with a better understanding of how different processors stack up against each other in terms of performance, architecture, and other key specifications.
  - This feature will require implementing advanced front-end interactivity, such as checkboxes for processor selection and dynamic comparison tables.

## 2. Expanded Data Sources:

- To provide users with more comprehensive data, the platform can integrate additional benchmark scores from other well-known testing platforms. Adding real-world test scores, power consumption details, and additional metrics such as **GPU performance** or **AI capabilities** will further enhance the platform's value.
- Integration with API services from benchmark platforms or hardware manufacturers could also automate the data collection process, ensuring that the platform remains up-to-date.

## 3. User Accounts and Personalization:

- Implementing a **user authentication** system, allowing users to create accounts and save their preferred processors or comparisons, would significantly improve the user experience. Users could track their favorite processors, compare them, and receive notifications when new models are added or updated.
- This would also enable the possibility of **user-generated content**, such as reviews or ratings for processors, which could add an additional layer of interactivity.

## 4. Enhanced Data Visualization:

- Adding **charts** and **graphs** to the platform would help users better visualize the performance data, such as bar charts or line graphs comparing benchmark scores across processors. This could make comparisons more intuitive and visually appealing.
- An interactive chart that shows the **performance trend over time** or compares processors from different years would further enhance the platform's analytical capabilities.

## 5. Integration with E-commerce Platforms:

- To provide a seamless experience for users, integration with e-commerce platforms could be explored. Users could not only compare processors but also check availability, pricing, and reviews across different online stores.
- This would allow SocSpecs to serve as a **one-stop shop** for users interested in buying processors or smartphones, making the platform more commercially viable.

## 6. Performance Optimization for Large Datasets:

- As the number of processors and related data grows, the platform's back-end should be optimized for **faster querying** and **efficient data management**. This may involve implementing database indexing or using more advanced database solutions that handle large-scale data more efficiently.
- Caching and load balancing techniques can also be employed to ensure that the platform remains responsive, even as user traffic and data volume increase.

## 7. Mobile Application Development:

- While the platform is currently mobile-responsive, developing a **native mobile application** could further enhance the user experience. A mobile app could allow for offline browsing, push notifications, and a more streamlined interface for users on the go.



# **Chapter Eight**

## **Appendices**

## 8.1 Appendix A: Source Code

The source code for the SocSpecs platform is structured to ensure clarity, maintainability, and modularity. Below is an overview of the primary components and their responsibilities:

/page.tsx (root)

```
import PsLogo from '@app/ui/acme-logo';
import { ArrowRightIcon } from '@heroicons/react/24/outline';
import Link from 'next/link';
import { lusitana, roboto } from '@app/ui/font';
import Image from 'next/image';
export default function Page() {
  return (
    <main className="flex min-h-screen flex-col p-6">
      <div className="flex h-20 shrink-0 items-end rounded-lg bg-blue-500 p-4 md:h-20 shadow-xl">
        { <PsLogo /> } <div className="flex flex-col">

          </div>
        </div>
        <div className="mt-4 flex grow flex-col gap-4 md:flex-row">
          <div className="flex flex-col justify-center gap-6 rounded-lg bg-gray-50 px-6 py-10 md:w-2/5 md:px-20">

            <p className={` ${roboto.className} text-xl text-gray-800 md:text-3xl md:leading-normal`}>
              <strong>Welcome to SocSpecs, </strong> the ultimate destination for comprehensive specifications and in-depth comparisons of System on Chips (SoCs)
            </p>
            <Link
              href="home/ranking"
              className="flex items-center gap-5 self-start rounded-lg bg-blue-500 px-6 py-3 text-sm font-medium text-white transition-colors hover:bg-blue-400 md:text-base"
            >
              <span>View Ranking</span> <ArrowRightIcon className="w-5 md:w-6" />
            </Link>
          </div>
          <div className="flex items-center justify-center p-6 md:w-3/5 md:px-28 md:py-12">
            <Image
              src="/soc-home.png"
              width={700}
              height={700}
              className="hidden md:block"
              alt="Screenshots of the dashboard project showing desktop version"
            />
          </div>
        </div>
      </div>
    </main>
  );
}
```

```

        <Image
          src="/soc-home.png"
          width={500}
          height={500}
          className="block md:hidden"
          alt="Screenshots of the dashboard project showing desktop version"
        />
      </div>
    </div>
  </main>
);
}

/home/layout.tsx (home)

import SideNav from '@app/ui/home/sidenav';
export const experimental_ppr = true;
export default function Layout({ children }: { children: React.ReactNode }) {
  return (
    <div className="flex h-screen flex-col md:flex-row md:overflow-hidden">

      <div className="w-full flex-none md:w-64">

        <SideNav />
      </div>

      <div className="flex-grow p-6 md:overflow-y-auto md:p-12">{children}</div>
    </div>

  );
}

```

/home/ranking/page.tsx (ranking)

```
import Table from '@app/ui/ranking/table';
import { lusitana, roboto } from '@app/ui/font';
import { RankingTableSkeleton } from '@app/ui/skeletons';
import { Suspense } from 'react';
import { fetchRanking } from '@app/lib/data';

export default async function Page() {

  const currentPage = 1;

  return (
    <div className="w-full">

      <div className="flex w-full items-center justify-between">
        <h1 className={` ${roboto.className} text-2xl`} >Smartphone Processors Ranking</h1>
      </div>

      <div className="mt-4 flex items-center justify-between gap-2 md:mt-8">
        <p className={` ${roboto.className} text-1xl`} >Updated performance rating. Click on the
name to see more detailed information about a particular chip.</p>
      </div>
      <Suspense fallback={<RankingTableSkeleton />}>
        <Table />
      </Suspense>
      <div className="mt-5 flex w-full justify-center">

        </div>
      </div>
    );
  }
```

/home/soc/[id]/page.tsx (soc[id])

```
'use client'
import { lusitana, roboto } from '@app/ui/font';
import { ProcessorDetails } from '@app/lib/definitions';
import { fetchDeviceDetails } from '@app/lib/data';
import { useParams } from 'next/navigation'; // Changed from next/router
import { useEffect, useState } from 'react';
import { DetailsSkeleton } from 'app/ui/skeletons'
import { Suspense } from 'react';
export default function Page() {
  const [details, setDetails] = useState<ProcessorDetails | null>(null);
  const { id } = useParams(); // Using useParams instead of useRouter

  useEffect(() => {
    console.log(id)
    if (id) {
      fetchDeviceDetails(id as string) // Assuming `id` is a string that needs to be converted
        .then((data) => setDetails(data[0])) // Assuming the result is an array, take the first
        element
        .catch((error) => console.error('Error fetching device details:', error));
    }
  }, [id]); // Re-run the effect when `id` changes
  <Suspense fallback={<DetailsSkeleton />}>

  </Suspense>
  if (!details) {
    return (
      <div className="flex flex-col justify-center rounded-lg bg-gray-50 px-2 py-10 md:px-
20">
        <DetailsSkeleton />
      </div>
    );
  }

  console.log(details)

  return (
    <div className="flex flex-col justify-center rounded-lg bg-gray-50 px-2 py-10 md:px-
20">
      <h1 className={` ${roboto.className} text-xl text-gray-800 md:leading-normal`} >
        {details.processor_name}
      </h1>
    </div>
  )
}
```

```

<div className="mt-6">
  <h2 className={` ${roboto.className} text-lg text-gray-600`} >Processor
Specifications</h2>
  <div className="mt-4 space-y-2">
    <p><strong>Antutu Score:</strong> {details.antutu_score}</p>
    <p><strong>Geekbench Single-Core Score:</strong>
{details.geekbench_singlecorescore}</p>
    <p><strong>Geekbench Multi-Core Score:</strong>
{details.geekbench_multicorescore}</p>
  </div>
</div>

<div className="mt-6">
  <h2 className={` ${roboto.className} text-lg text-gray-600`} >CPU
Specifications</h2>
  <div className="mt-4 space-y-2">
    <p><strong>Architecture:</strong> {details.cpu_architecture}</p>
    <p><strong>Primary Core:</strong> {details.cpu_primarycore}</p>
    <p><strong>Secondary Core:</strong> {details.cpu_secondarycore}</p>
    <p><strong>Efficiency Core:</strong> {details.cpu_efficiencycore}</p>
    <p><strong>Total Cores:</strong> {details.cpu_totalcores}</p>
    <p><strong>Frequency:</strong> {details.cpu_frequency}</p>
    <p><strong>Instruction Set:</strong> {details.cpu_instructionset}</p>
    <p><strong>Manufacturing Process:</strong> {details.cpu_process}</p>
    <p><strong>Manufacturing Company:</strong> {details.cpu_manufacturing}</p>
  </div>
</div>

<div className="mt-6">
  <h2 className={` ${roboto.className} text-lg text-gray-600`} >Graphics
Specifications</h2>
  <div className="mt-4 space-y-2">
    <p><strong>GPU Name:</strong> {details.graphics_gpuname}</p>
    <p><strong>Graphics Architecture:</strong> {details.graphics_architecture}</p>
    <p><strong>Graphics Frequency:</strong> {details.graphics_frequency}</p>
    <p><strong>Graphics Pipelines:</strong> {details.graphics_pipelines}</p>
    <p><strong>Shading Units:</strong> {details.graphics_shadingunits}</p>
    <p><strong>Total Shaders:</strong> {details.graphics_totalshaders}</p>
    <p><strong>FLOPS:</strong> {details.graphics_flops}</p>
    <p><strong>Vulkan Version:</strong> {details.graphics_vulkanversion}</p>
    <p><strong>OpenCL Version:</strong> {details.graphics_openclversion}</p>
    <p><strong>DirectX Version:</strong> {details.graphics_directxversion}</p>
  </div>
</div>

<div className="mt-6">

```

```

        <h2 className={`$ ${roboto.className} text-lg text-gray-600`} >General
Information</h2>
        <div className="mt-4 space-y-2">
            <p><strong>Announced Date:</strong> {details.announced_date}</p>
            <p><strong>Manufacturing Process:</strong>
{details.manufacturing_process}</p>
            <p><strong>Manufacturing Company:</strong>
{details.manufacturing_company}</p>
            <p><strong>Info Class:</strong> {details.info_class}</p>
        </div>
    </div>
</div>
);
}

```

/app/lib/definitions.ts

```
export type Ranking = {  
  rank: number;  
  name: string;  
  antutu_score: number;  
  geekbench_singlecore: number;  
  geekbench_multicore: number;  
  threedmark_steelnomad_score: number;  
  image_url: string;  
  uuid: string;  
}
```

```
export type ProcessorDetails = {  
  id: number;  
  processor_name: string;  
  antutu_score: number;  
  geekbench_singlecorescore: number;  
  geekbench_multicorescore: number;  
  
  cpu_architecture: string;  
  cpu_primarycore: string;  
  cpu_secondarycore: string;  
  cpu_efficiencycore: string;  
  cpu_totalcores: number;  
  cpu_frequency: string;  
  cpu_instructionset: string;  
  cpu_process: string;  
  cpu_manufacturing: string;  
  
  graphics_gpuname: string;  
  graphics_architecture: string;  
  graphics_frequency: string;  
  graphics_pipelines: number;  
  graphics_shadingunits: number;  
  graphics_totalshaders: number;  
  graphics_flops: string;  
  graphics_vulkanversion: string;  
  graphics_openglversion: string;  
  graphics_directxversion: string;  
  
  announced_date: string;  
  manufacturing_process: string;  
  manufacturing_company: string;  
  info_class: string;  
  ranking_uuid: string;  
};
```



/app/lib/data.ts

```
"use server"
import { sql } from '@vercel/postgres';
import {
  Ranking,
  ProcessorDetails
} from './definitions';
import { formatCurrency } from './utils';

export async function fetchRanking() {
  try {
    const data = await sql<Ranking>`SELECT * FROM rankings ORDER BY rank ASC`;
    return data.rows;
  } catch (error) {
    console.error('Database Error:', error);
    throw new Error('Failed to fetch ranking data.');
  }
}

export async function fetchDeviceDetails(id: string) {
  try {
    const data = await sql<ProcessorDetails>`SELECT * FROM processordetails WHERE
ranking_uuid = ${id}`;
    return data.rows;
  } catch (error) {
    console.error('Database Error:', error);
    throw new Error('Failed to fetch device details.');
  }
}
```

## 8.2 Appendix B: Database Queries

Below are the SQL queries used to insert and retrieve data in the SocSpecs platform:

### Inserting processor details:

```
INSERT INTO processors (  
    processor_name, antutu_score, geekbench_singleCoreScore,  
    geekbench_multiCoreScore,  
    cpu_architecture, cpu_primaryCore, cpu_secondaryCore, cpu_efficiencyCore,  
    cpu_totalCores, cpu_frequency, cpu_instructionSet, cpu_process,  
    cpu_manufacturing, graphics_gpuName, graphics_architecture,  
    graphics_frequency, graphics_pipelines, graphics_shadingUnits,  
    graphics_totalShaders, graphics_flops, graphics_vulkanVersion,  
    graphics_openCLVersion, graphics_directXVersion, announced_date,  
    manufacturing_process, manufacturing_company, info_class  
)  
  
VALUES  
(  
    'Qualcomm Snapdragon 8 Elite (Gen 4)', 3109010, 3234, 10059,  
    'Oryon (Phoenix L)', '2x 4.32 GHz – Oryon (Phoenix L)', '6x 3.53 GHz – Oryon  
(Phoenix M)',  
    NULL, 8, '4320 MHz', 'ARMv9.2-A', '3 nanometers', 'TSMC',  
    'Adreno 830', 'Adreno 800', '1100 MHz', 2, 1536, 3072,  
    '3379.2 GigaFLOPS', '1.3', '3.0', '12.1', '2024-10-21', '3 nanometers', 'TSMC',  
    'Flagship');
```

### Retrieving rankings:

```
SELECT * FROM rankings ORDER BY rank ASC;
```

### 8.3 Appendix C: Wireframes

The following wireframes illustrate the layout of key pages within the SocSpecs platform:

- **Landing Page:** Includes a processor image, a welcome text section, and the "View Ranking" button.
- **Ranking Page:** Displays the processor ranking table with columns for rank, processor name, and benchmark scores.
- **Processor Details Page:** Shows detailed specifications of a processor, including CPU, GPU, and general information.

### 8.4 Appendix D: User Feedback

User feedback was collected through a series of surveys and beta testing, and the following points were noted:

- **Positive Feedback:**
  - Users found the interface intuitive and easy to navigate.
  - The ranking table and processor details were well-received for their clarity and accuracy.
  - The mobile responsiveness of the platform was appreciated.
- **Suggested Improvements:**
  - Users expressed a desire for the **Compare Processors** feature to be implemented.
  - Some users suggested adding more data on **power consumption** and **real-world performance**.

## 8.5 Appendix E: Reference

1. A. Stevens, *C++ Database Development*, MIS Press, New York, 1992, p. 34.
2. D.L. Carney, J.I. Cochran, "The 5ESS Switching System: Architectural Overview," *AT&T Technical Journal*, vol. 64, no. 6, July-August 1985, pp. 1339-1356.
3. J. Martin, *Computer Database Organization*, Prentice-Hall, Englewood Cliffs, NJ, 1977, p. 53.
4. M. Smith, "Introduction to Full-Stack Development with Next.js and PostgreSQL," *O'Reilly Media*, 2021.
5. S. Patel, "Tailwind CSS for Responsive Web Design," *Web Development Journal*, vol. 12, no. 5, 2022, pp. 45-50.
6. PostgreSQL Global Development Group, "PostgreSQL Documentation," [Online]. Available: <https://www.postgresql.org/docs/>.
7. Vercel, "Next.js Documentation," [Online]. Available: <https://nextjs.org/docs>.
8. Tailwind Labs, "Tailwind CSS Documentation," [Online]. Available: <https://tailwindcss.com/docs>.
9. D. Williams, "Optimizing Performance with PostgreSQL for Full-Stack Applications," *Database Solutions Journal*, vol. 7, no. 4, 2023, pp. 12-19.
10. D. Clark, "Creating Scalable Web Applications with Next.js," *Tech Review*, vol. 8, no. 3, 2024, pp. 22-29.
11. M. Johnson, "Processor Benchmarking and Comparison," *TechInsights Magazine*, vol. 5, no. 2, 2023, pp. 11-17.
12. Qualcomm, "Snapdragon Processor Specifications," [Online]. Available: <https://www.qualcomm.com/products/snapdragon>.
13. S. Lee, "Building Modern Web Applications with TailwindCSS and Next.js," *Full-Stack Developer's Journal*, vol. 10, no. 1, 2024, pp. 100-105.

15. Coursera, "Online Courses and Degrees," [Online]. Available: <https://www.coursera.org>.
16. Udemy, "Online Learning and Courses," [Online]. Available: <https://www.udemy.com>.
17. "Next.js Dashboard App: Getting Started," [Online]. Available: <https://nextjs.org/learn/dashboard-app/getting-started>.
18. Vercel, "Vercel - Next.js Deployments," [Online]. Available: <https://vercel.com/>.