**Name: Aleeza Qadeer**

**Registration No: SP21-BSE-033**

***Operating System*:**

**https://github.com/xomboverlord/xomb**

**Description:**

* XOmB is an exokernel-based operating system, designed to give applications direct access to hardware, minimizing the OS’s intervention. Unlike traditional monolithic kernels, it only handles basic functions like memory allocation and process protection, letting applications implement higher-level operations. This results in high flexibility and control, ideal for custom environments
* XOmB, or "XOmB," is an experimental exokernel operating system developed to allow applications more direct and fine-grained control over hardware. Exokernels, unlike traditional monolithic or microkernels, provide minimal abstractions. They offload higher-level operations (like file systems or network protocols) to user space libraries, giving developers the freedom to implement or optimize their own system-level operations. XOmB specifically is built with the D programming language, making it unique in its lightweight design and aiming to work efficiently on 64-bit multi-core systems. The system provides the essentials to allocate and protect resources, while higher-level OS functionalities are left to user space.

**Key Components of XOmB:**

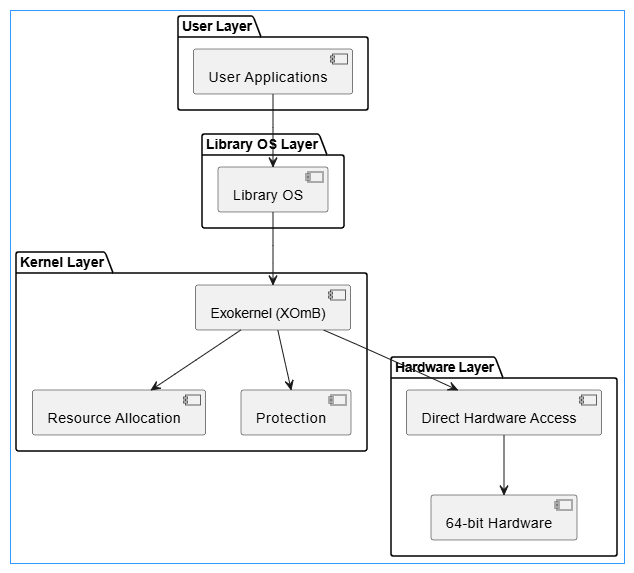
1. **Kernel**:
   * The core of XOmB is a minimalistic kernel responsible primarily for resource management (e.g., memory allocation, processor time) and ensuring application isolation.
   * The kernel includes low-level handlers for system calls and manages permissions, allowing applications to access hardware directly while enforcing security where necessary.
2. **Library OS**:
   * Unlike in monolithic systems, where system calls are deeply embedded, XOmB uses a "Library OS" model.
   * Applications can bring in libraries that replicate typical OS functionalities (e.g., file management, device handling) tailored to their needs.
   * This approach allows developers to customize these libraries, optimizing them for specific hardware or performance requirements.
3. **Hardware Interface**:
   * XOmB interacts directly with hardware, providing basic drivers and interfaces at the kernel level.
   * The hardware interface in the kernel supports 64-bit multi-core systems, and users can add custom modules to support additional hardware if needed.
4. **User Applications**:
   * Applications interact with the kernel through the library OS layer, which simplifies hardware access and adds functionalities as required.
   * Due to the exokernel design, applications must handle certain low-level functionalities, which allows fine-grained control but also requires more custom setup.

**Interaction Model:**

The interaction in XOmB is direct yet structured. Applications communicate through the Library OS to request resources from the kernel. The kernel allocates these resources without adding OS-level abstractions, allowing applications to access hardware with minimal overhead. This model benefits advanced users needing maximum control and minimal latency.

**Visualization Diagram**

Below is a conceptual diagram illustrating XOmB's structure:



**Compilation and Setup:**

To compile XOmB:

1. Clone the repository from GitHub.
2. Ensure you have a D language compiler (such as dmd or ldc2) and necessary build tools.
3. Follow the repository’s README for environment setup, dependencies, and specific build commands.

**Suggestions for Improvement:**

1. **Enhanced Hardware Support**: Expanding drivers and interfaces for more hardware types would make XOmB more versatile.
2. **Expanded Documentation**: More comprehensive guides for developers could improve usability and support.
3. **User-space Libraries**: Providing pre-built Library OS components for common tasks (file handling, network protocols) could streamline development.