BMO OS - Project Management

1. Project Management

2.1 Project Overview

BMO OS is a custom, modular Linux-based operating system built completely from source using **Linux From Scratch (LFS)** and **Beyond Linux From Scratch (BLFS)** methodologies.

It is designed to provide a **lightweight, developer-friendly**, and **AI-ready** environment.

The system will later integrate **Flutter** for a modern desktop experience and **local AI features** to enable intelligent offline tools such as code assistants, speech recognition, and system automation.

1.2 Project Objectives

- Build a stable, bootable Linux-based operating system.
- Document and understand all low-level system components (kernel, libraries, network stack).
- Develop a **UI layer** as the visual system shell.
- Support **SSH**, **network configuration**, and **software management**.
- Add **Local AI capabilities** for offline intelligence and automation (future phase).
- Package the OS as a **bootable ISO** for testing and distribution.

1.3 Project Scope

Included:

- Core system (kernel, init system, toolchain, utilities).
- Networking and SSH configuration.
- Bootloader setup (GRUB2) and ISO generation.

- Flutter-based / KDE graphical shell.
- Local AI engine integration (future phase).

Not included (initial release):

- Full package manager (planned later).
- Cloud AI or networked inference (only local).
- Advanced desktop environment (beyond UI).

1.4 Project Development Model

Incremental + Modular Development Model

The OS will be developed in incremental phases where each phase delivers a working system layer. Future modules like **Local AI** will be added as self-contained components.

Phase	Description	Deliverables
Phase 1	LFS Core System	Basic Linux environment with CLI
Phase 2	BLFS Networking & SSH	Network stack and remote access
Phase 3	Bootloader & ISO	GRUB setup and bootable ISO
Phase 4	Desktop UI	System interface layer
Phase 5	Testing & Optimization	Final system validation
Phase 6	**Local AI Integration**	AI module for offline smart features

---

1.5 Local AI Integration Plan (for future if we decide to continue)

Aspect	Description
Goal	Integrate an offline AI subsystem to enhance the OS with intelligent local tools.
Examples	Local chatbot, code assistant, command predictor, voice-based control, and context-aware notifications.
Technology Stack	Python (AI engine), ONNX Runtime / GGML / llama.cpp for model inference, Flutter frontend for interaction.
Data Privacy	100% local processing — no external cloud access.

Aspect	Description
Hardware Needs	Minimum 4 GB RAM, x86_64 CPU with AVX2 (or ARM64).
Integration Level	Runs as a system daemon (`bmocore-ai`) communicating via sockets or REST API with UI layer.

1.6 Project Roles and Responsibilities

Role	Name
Project Manager & Lead Developer	Abdelrahman Ashraf
System Architect	Walled Emad
UI Drawer	Abdelrahman
UI Devs	Abdelrahman Ashraf - Abdelrahman - Saif Mohamed
Tester & Documentation Manager	Omar Salah - Saif Mohamed - Mohamed

1.7 Project Tools and Environment

Category	Tools / Environment
Host OS	Fedora Linux / ubuntu linux
Build Tools	GCC, Binutils, Make, CMake, Meson, Ninja
Programming Languages	C, C++, Bash, Python, Dart
UI Framework	Flutter / Qt
Version Control	Git + GitHub
Boot Tools	GRUB2, Xorriso
Virtualization	QEMU, VirtualBox
Editors	Vim, VS Code , nano
Testing	GDB, DejaGNU
AI Frameworks (Planned)	//

1.8 Project Timeline (Revised)

Phase	Duration	Status
Phase 1 – LFS Base System	3 Weeks	✅ Completed
Phase 2 – BLFS Networking & SSH	2 Weeks	🟢 In Progress
Phase 3 – Bootloader & ISO	1 Week	🕓 Planned
Phase 4 – UI Layer	3 - 4 Weeks	🕓 Planned
Phase 5 – Testing & Optimization	2 Weeks	🕓 Planned
Phase 6 – Local AI Integration	**5 Weeks**	🕓 Future Development

1.9 Risk Management

Risk	Description	Mitigation
AI Performance Issues	Local inference may consume heavy RAM/CPU	Use quantized models (GGUF format)
Security Risks	AI modules running with system privileges	Use sandboxing and permission limits
Build Failures	Errors in compiling packages	Keep build logs and snapshots
Boot Issues	GRUB or kernel misconfiguration	Test each step with QEMU
Time Overruns	Manual building takes time	Follow modular milestone schedule

1.10 Communication and Reporting

All progress, commits, and build notes are stored in the **GitHub repository**.

Documentation is maintained in:

- `README.md` – Build steps and progress
- `CHANGELOG.md` – Version history
- `docs/` folder – SRS, SDS, AI integration plan

1.11 Project Success Criteria

The project will be considered successful when:

1. The OS boots successfully and runs stable.
2. Networking and SSH are functional.
3. KDE UI operates as the default desktop environment.
4. The system can run and interact with a local AI module offline (not for the first release patch).
5. A bootable ISO is generated and tested successfully on real and virtual hardware.