my minimum requirement aka the target device is of 1gb ram and processor older than 2020, however any phone after that must also be able to install the os

i think any android based after 4-5 must be able to install

play capability should be side loaded via user setup

youtube will also be given option to load an app or go variant or things like that

a lightweight chromium based browser is better

our os will have a single unified lightweight aurdio and video player

we will think about camera later

all the sensors present and working must be supported in the os

yes dialer and messaging must be availabe in the phone

basic contacts and offline calandars will be included

os will have only a few wdgets and just minimal look, it will have 2 themes dark and light, and ui will only have black and white colors, however ui will change as per the application used

default notifications

background apps will be setted by users apart from some applications like clock, calandar, sms, calls etc

yes data saver will always be turned on

there will be no or little animations in the phone

yes minimum system partition size target will be there

minimum ram requirement will be 1gb

power comsumption must be minimal

allow donloadable keyboard, but have one keyboard like g-board pre installed

we will think about accessibility later

no no parental control needed this os is not for kids

we will think about updates later

location is only turned on when needed

bluetooth and wifi are available

user can install any app without any problem, as long as user trust the author of the app

minimal setup with few toggles

yes include feedback/ log collection in order to keep no user facing crash UI

now give me list of all the features of my os named "dummyOS" and also give me an entire roadmap on how to build this proejct
also list the technologies needed in order to build thi

Direct answer: dummyOS will be an Android-Go–tuned, minimal, privacy-lean system targeting 1 GB RAM devices and older SoCs, shipping a lightweight launcher, unified media player, basic telephony/SMS/contacts/calendar, a Chromium-based browser, optional Play and YouTube via side-load, strict background controls, data saver on by default, black/white UI with light/dark themes, and user-controlled app installs, with sensors enabled when present. Below is a complete feature list, technology stack, and an end-to-end roadmap from device selection to release and maintenance.

Core feature set

- Target hardware: guaranteed support for devices with **1 GB RAM** and SoCs predating 2020, with broader compatibility up to newer devices; installs allowed on later devices as well.[1] [2]
- OS base: Android-Go configuration of AOSP with low-RAM policies, tuned services, and reduced footprint to meet performance on constrained devices.[3] [1]
- App model: default minimal app set; Play capability available via side-load during first-run or Settings, not preinstalled; YouTube offered as official, Go, or web/PWA choice.[4] [1]
- UI/UX: custom minimal launcher, black-and-white palette with two themes (light/dark), few widgets (e.g., clock, battery), almost no animations for responsiveness.[1]
- Communications: phone dialer, SMS/MMS messaging, contacts, and offline calendar bundled as lightweight system apps.[1]
- Browser: lightweight Chromium-based browser using System WebView, tuned for data saver and low memory.[4] [1]
- Media: single unified lightweight audio/video player with basic playlists and hardware-accelerated playback when codecs permit.[1]
- Sensors: enable all available device sensors (accelerometer, magnetometer/compass, proximity, light, gyro if present) for system and app use.[3]

- Power/data: Data Saver always on; aggressive power policies by default while keeping telephony reliability; location toggled on-demand only.[3] [1]
- Background control: essential apps (clock/alarms, calendar, SMS, calls) protected; user config controls for other apps' background execution; overall low-RAM limits enforced.[3]
- Input: one preinstalled keyboard (Gboard-class) with downloadable packs; minimal languages built-in, others downloadable.[1]
- Connectivity: Wi-Fi and Bluetooth supported; tethering allowed.[1]
- Install policy: users can install any app they trust; no parental/child controls; sideloading enabled from Settings with warnings.[1]
- Crash handling: background crash capture and opt-in feedback/log collection; no intrusive crash UI presented to user.[1]
- Partitioning: defined minimum system partition size target; optimize preinstalled set to fit legacy devices; use Go app variants where applicable.[1]
- Accessibility and camera: deferred to later phases; OS plumbing prepared for future enablement.[1]

Technology stack

- Base OS: AOSP with Android-Go overlays and properties (e.g., ro.config.low_ram) for the chosen Android release that aligns with available device BSPs.[3] [1]
- Build/infra: repo/Soong/Make-based AOSP build; device trees, kernel source or prebuilts, and vendor blobs per device; OTA packaging optional in later phase.[5] [6]
- UI: custom launcher (Android app) with minimal feature set; system overlays for theming, fonts, colors (black/white), and animation scale reductions.[1]
- Web: Android System WebView or Chromium WebView shell for the lightweight browser; network/data-saver defaults.[1]
- Media: ExoPlayer-based unified media app with hardware decode paths; fallback software decode on older SoCs.[1]
- Telephony/SMS: AOSP Dialer and Messaging forks or light alternatives configurable to Go; RIL/HAL compatibility from device BSPs.[1]
- Sensors/location: Sensor HAL via device BSP; location providers constrained to on-demand; permissions defaults tightened.[3]
- Power/memory: LMKD thresholds tuned for 1 GB targets; zram enabled; reduced background service limits; job scheduler constraints and app standby tuned.[3]
- Security: signature-level permissions for system apps, verified boot aligned with device capability, SELinux in enforcing; install-from-unknown-sources flow preserved with warnings.[1]
- Feedback: background crash reporter using DropBoxManager/metricsd-style pipeline with opt-in toggle in Settings.[1]

Version and device targeting

- Android release selection: prioritize a version where device trees and blobs are readily available for 2016–2020-era devices (e.g., Android 11–13 Go) while maintaining forward compatibility for newer devices.[2] [1]
- Policy awareness: newer Android releases raise RAM/storage requirements for full Android; Go remains the path for ≤2–3 GB RAM devices, aligning with dummyOS's 1 GB baseline.[7] [8]
- Device selection: start with a small set of phones having open kernels/BSPs and widespread community ROM support to minimize bring-up time.[5]

End-to-end roadmap

- Phase 0 — Planning (1–2 weeks)
  - Lock the base Android version and Go edition level that matches the widest set of legacy devices yet still maintains modern security patches.[2] [1]
  - Choose 2–3 reference devices with available device/kernel/vendor trees and known working telephony and basic sensors.[5]
  - Define minimum system partition and app budget; outline allowed system apps and their sizes.[1]
- Phase 1 — Build environment and source sync (1–2 weeks)
  - Set up build host per current AOSP requirements; plan for high RAM usage—enable swap/zram and build with lower parallelism if needed.[6] [9]
  - repo init/sync AOSP; pull device trees, kernels, and vendor blobs for reference devices; confirm clean build of stock AOSP for sanity.[5]
- Phase 2 — Go enablement and product definition (2–3 weeks)
  - Create a product makefile for dummyOS enabling Android-Go/low-RAM flags and overlays (ro.config.low_ram=true, Go features).[3] [1]
  - Assemble minimal package list: launcher, browser, unified media, dialer, messaging, contacts, calendar, keyboard; remove heavy/nonessential services.[1]
  - Set global defaults: Data Saver on, reduced animation scales, strict background limits with exemptions for alarms, telephony, SMS, calendar.[3]
- Phase 3 — UI/UX shell and theming (2–4 weeks)
  - Implement the custom minimal launcher (black/white palette; light/dark themes; few widgets); expose quick toggles for Wi-Fi, Bluetooth, location, data saver.[1]
  - Apply system overlays for colors/typography and limit widgets/animations system-wide; ensure responsiveness on 1 GB devices.[1]
- Phase 4 — Core apps integration (3–5 weeks)
  - Browser: integrate lightweight Chromium WebView shell with data saver defaults; validate low-RAM behavior.[1]

    - Media: implement ExoPlayer-based unified audio/video player; validate hardware decode and fallbacks.[1]
    - Telephony/Messaging/Contacts/Calendar: integrate light variants; ensure SMS/MMS and calling flows; offline calendar storage.[1]
    - Keyboard: preinstall minimal keyboard and enable downloadable language packs.[1]
- Phase 5 — Sideload flows and optional services (2–3 weeks)
    - First-run or Settings flows to optionally side-load Play capability and user's choice of YouTube app or web; document and test install-from-unknown-sources path.[4] [1]
    - Ensure store-independent app installation works with user consent and warnings; verify signature installs and rollback handling.[1]
- Phase 6 — Power, memory, and sensor tuning (3–4 weeks)
    - Tune LMKD thresholds, zram size, background app caps, and job scheduler windows for 1 GB devices; validate app cold start and UI latency targets.[3]
    - Verify sensor HAL behavior and compass accuracy; ensure location is off by default and only enabled on demand.[3]
- Phase 7 — Stability, telemetry, and QA (3–6 weeks)
    - Add background crash reporter with opt-in toggle; collect anonymized logs for triage; keep the UI free of crash popups.[1]
    - Build test plans: telephony reliability, SMS/MMS, browser/video playback, standby drain, alarms, and low-RAM stress; run on all reference devices.[3] [1]
- Phase 8 — Packaging, flashing, and documentation (2–3 weeks)
    - Produce signed images; provide flashing scripts and guides per device; define minimum system partition sizes and known limitations.[5]
    - Publish user guide for minimal setup, sideloading Play/YouTube, background app settings, and privacy options.[1]
- Phase 9 — Public preview and feedback loop (ongoing)
    - Release preview images for selected devices; gather feedback from logs and user reports; iterate on performance and compatibility.[1]
    - Expand device support incrementally, prioritizing models with available kernels/blobs and active communities.[5]
- Phase 10 — Future enhancements (backlog)
    - Camera pipeline enablement and optimization per device; accessibility features; OTA/A-B updates; broader app allowlist tooling and parental options if ever needed.[1]

Key policies and defaults

- Performance: strict low-RAM mode, zram enabled, minimal animations, and foreground priority to essentials for smoothness on 1 GB RAM.[3]
- Privacy and data: Data Saver on by default; location off by default, on-demand only; user-controlled app background privileges.[3] [1]

- Compatibility: use Android-Go–optimized components to maintain modern behavior on legacy hardware while allowing installs on newer devices.[2] [1]

Risks and mitigations

- Vendor HAL gaps: older devices may have missing/buggy camera or sensor HALs—stage camera later and publish device support notes.[5]
- Storage/RAM requirements evolving: new Android versions raise minimums—pin supported base versions per device class, keep Go path for ≤2 GB.[8] [7]
- Performance regressions: maintain continuous performance testing on 1 GB RAM reference hardware with strict acceptance thresholds.[3]

If helpful, the next step can be to lock two specific reference devices and select the exact Android-Go base version to maximize HAL availability and minimize backporting effort.[5] [1]

⁂

1. https://developer.android.com/guide/topics/androidgo
2. https://simple.wikipedia.org/wiki/Android_Go
3. https://android.googlesource.com/platform/docs/source.android.com/+/bb6e1eb86b31dd2d824a3639d7994dac8dffa9e4/en/devices/tech/perf/low-ram.html
4. https://en.wikipedia.org/wiki/Android_Go
5. https://source.android.com/docs/setup/start
6. https://source.android.com/docs/setup/start/requirements
7. https://www.heise.de/en/news/Andoid-15-Google-raises-minimum-hardware-requirements-10352631.html
8. https://www.gsmarena.com/here_are_googles_new_minimum_ram_and_storage_requirements_for_android_phones-news-67387.php
9. https://stackoverflow.com/questions/79037240/is-it-possible-to-build-aosp-on-32gb-ram
10. https://www.privacyportal.co.uk/blogs/free-rooting-tips-and-tricks/everything-you-need-to-know-about-android-go-phones
11. https://www.android.com/intl/en_in/versions/go-edition/
12. https://support.forcemanager.net/en/articles/8613604-what-are-the-minimum-requirements-for-android-devices
13. https://bigohtech.com/android-go-vs-regular-android