Dynamic Theme Manager for GNUstep

Presented by: Kiarash Mirkamandari and Ebrahim Haghshenas

Group Leader: Masih Hashemi

All members:
Seyed Ebrahim Haghshenas (abstract, intro, architecture, presentation)
Masih Hashemi (architecture, slides)
Kiarash Mirkamandari (architecture, slides, presentation)
Bahar Rahimivarposhti (architecture, use cases)
Kiarash Soleimani Roozbahani (architecture, control and data flow)
Andrew Ternopolsky (diagrams)

Agenda

1. Motivation & Rationale
2. Current Architecture
3. Proposed Dynamic Theme Manager
4. Two Approaches (SAAM Overview)
5. Sequence Diagram (Use Case)
6. Impact & Testing
7. Limitations & Lessons Learned
8. Conclusion

Motivation & Rationale

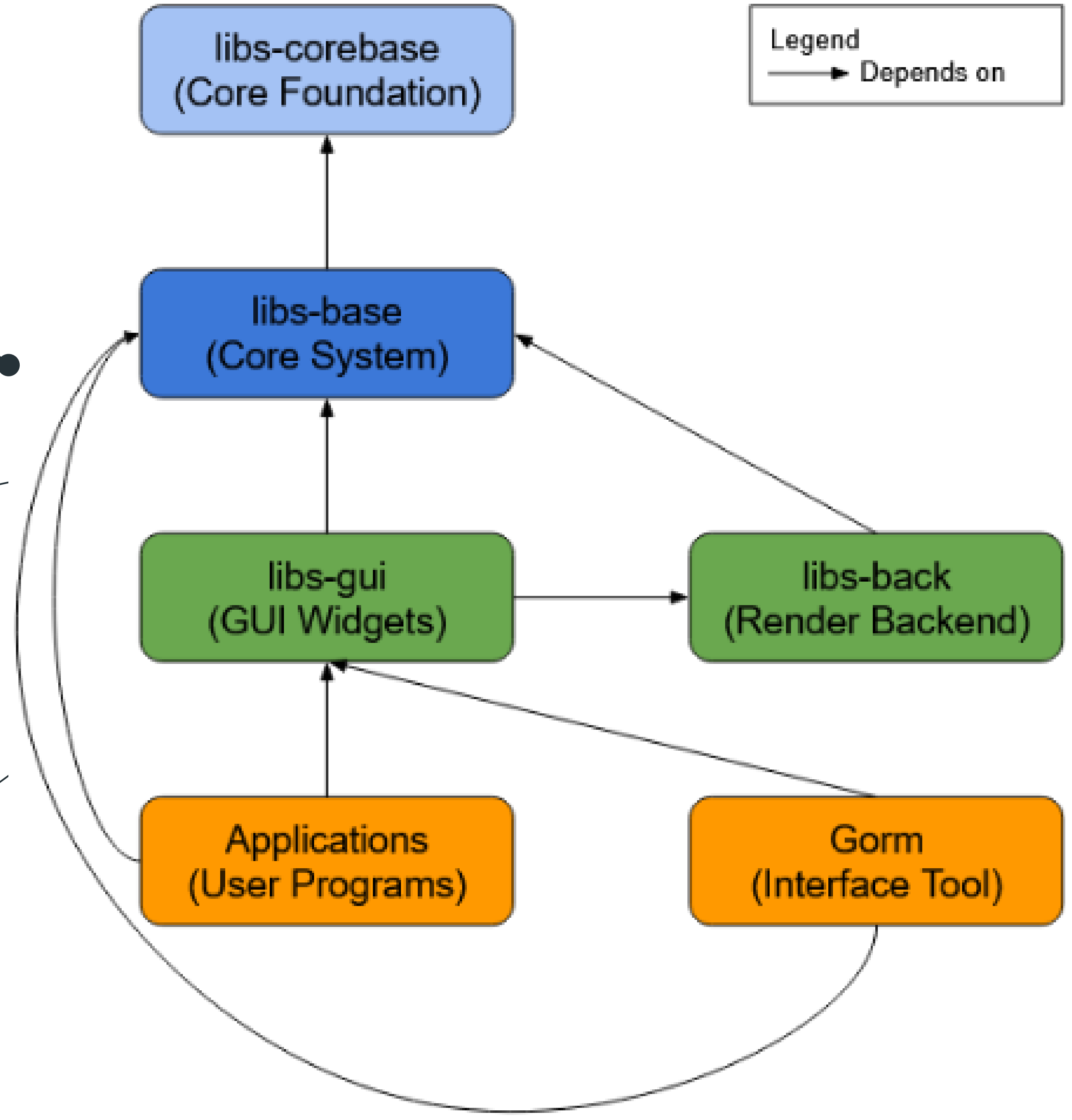
Core Need	Stakeholders	Driver
• Modern User Expectations for Runtime Theming	• Developers, end-users, managers	• Accessilibity + UI Consistency

Current Architecture

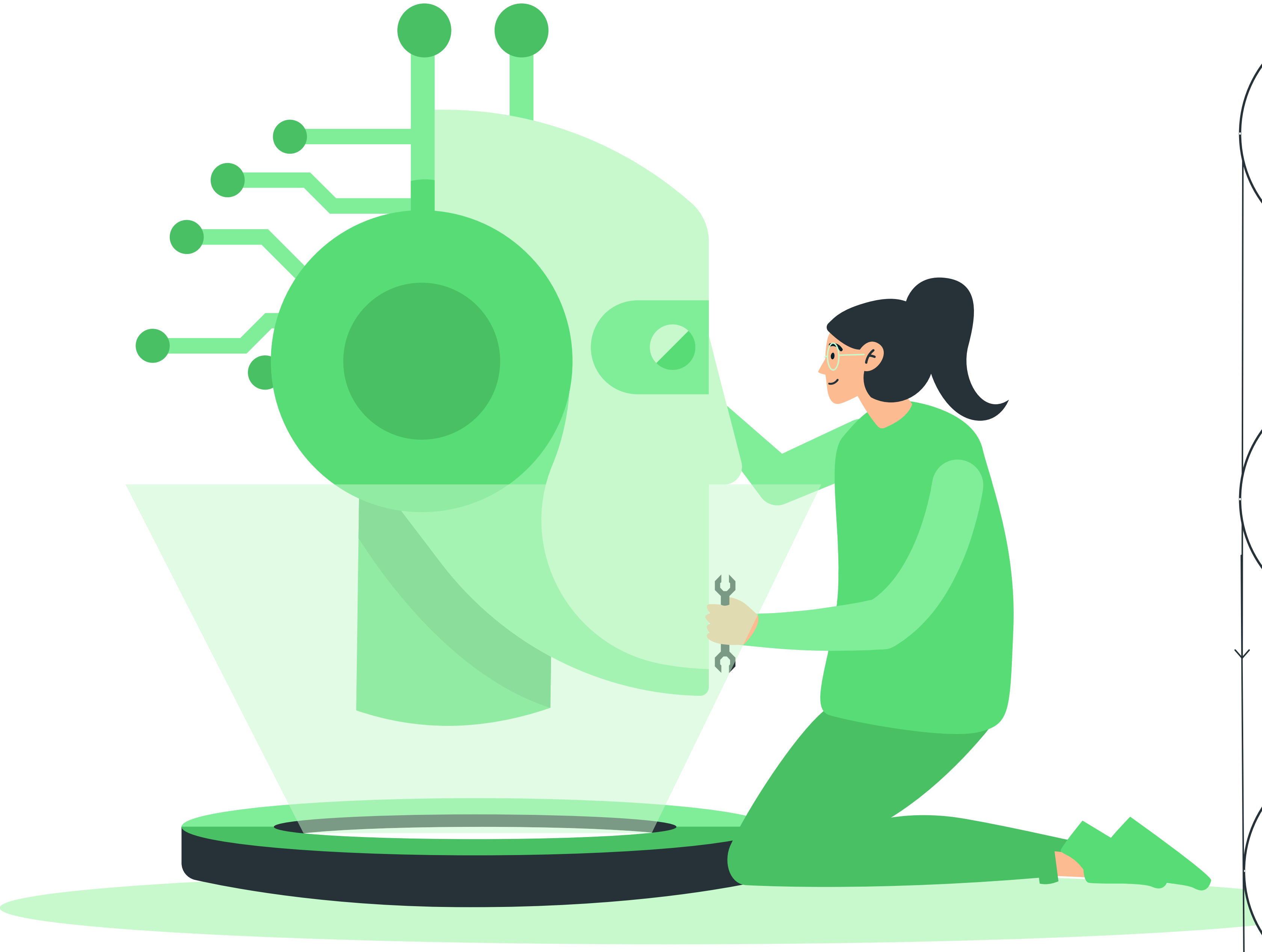
GNUstep is an open-source implementation of Apple’s OpenStep/Cocoa

Key conceptual layers: libs-corebase, libs-base, libs-gui, libs-back, libobjc2.

Originally assumed strict layering and single backend in A1.



Proposed Dynamic Theme Manager



Key Role: Centralizes theme data & broadcasts updates

Where It Fits: Extends libs-gui or separate module (libs-theme)

High-Level Benefit: Runtime theme switching, minimal disruption

Two Approaches (SAAM Overview)

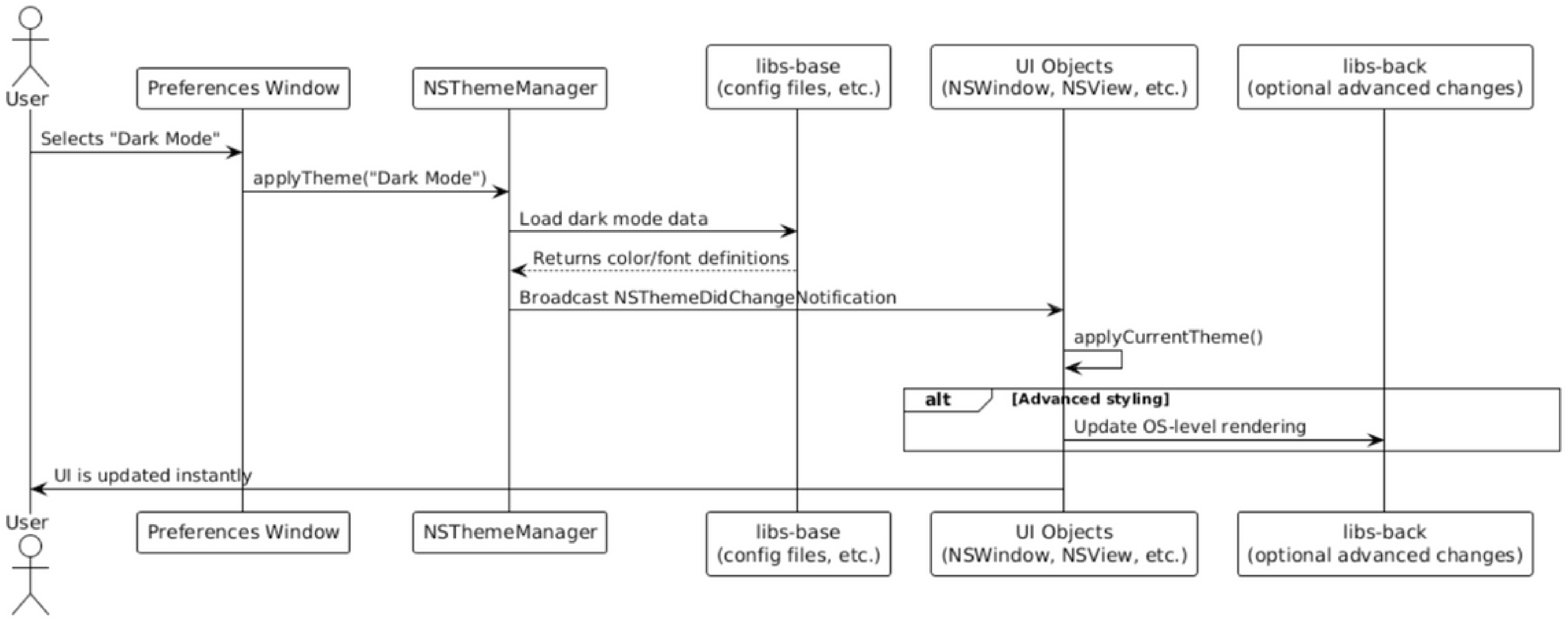
Approach A: Core Integration

Faster calls, - Invasive changes

Approach B: Plugin-Based

Modular & testable, - Slight overhead

Sequence Diagram (Use Case)



Use Case: User switches to Dark Mode

1 User → Preferences

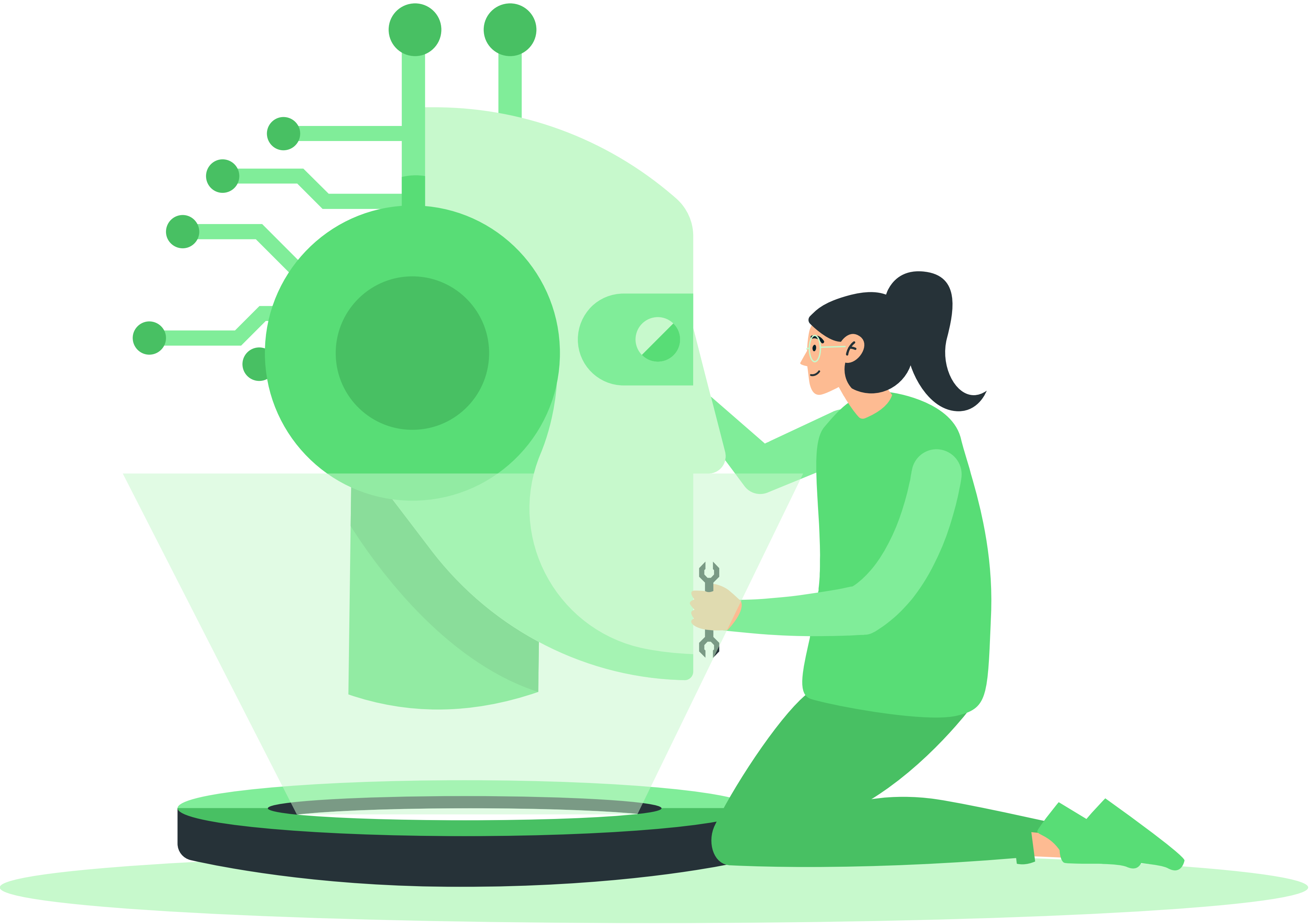
2 Theme Manager loads theme

3 Broadcast update → UI refresh

Impact & Testing

Impacted Subsystems	Concurrency	Testing Plan
• libs-base (config I/O), libs-gui (UI changes), libs-back (render tweaks)	• Main-thread UI + potential background loading	• Unit + integration + visual regression checks

Limitations & Lessons Learned



Limitations: OS mismatch, frequent switching overhead

Lesson: Notification-based updates fit well with Objective-C

Lesson: Modular design = safer, but watch for performance trade-offs

Conclusion

1

Enhances user experience with dynamic theming

2

Two possible designs with SAAM trade-offs

3

Helps future-proof GNUstep’s UI