GROUP#8: 500-INTERNAL SERVER ERROR

Proposal for Mobile Integration

https://youtu.be/cmV4aLM8nrI

GROUP MEMBERS

- Qiu Xing (Nathan) Cai - 21qxc@queensuca (Presenter)
- Haoxi Yang - 21hy11@queensu.ca (Presenter)
- Yuda Hu - 21yh29@queensu.ca
- Nicholas Wang - 22nw3@queensu.ca
- Amethyst Shen - 21YC121@QueensU.ca
- Wanting Huang - 20wh18@queensu.ca (Group Leader)

CONTENTS LIST

Introduction & Motivation

- GNUstep is a mature Objective-C framework—but **desktop-only**.
- No support for **mobile OSes** like iOS or Android:
  - No rendering for mobile APIs
  - No touch or gesture input
  - No lifecycle management (pause/resume/etc.)
- Why mobile?
  - Most users now access software via mobile devices.
  - Developers want cross-platform capabilities from a unified codebase.
  - Open-source frameworks (e.g., React Native, Flutter) have set modern standards.

CURRENT ARCHITECTURE

- Component Layer
  - Libs-base
  - Libs-corebase
  - Libs-gui
  - Libs-back
  - Gorm

- Layered Architecture

Current Limitations

- `libs-back` tied to desktop rendering
- No support for mobile input
- Missing mobile lifecycle handling
- No compatibility with mobile screen layouts or power constraints
- Porting to iOS/Android requires major changes

PROPOSED ENHANCEMENT

- Rendering Backends:
  - iOS: Core Graphics (Quartz 2D)
  - Android: SurfaceFlinger
- Touch Input:
  - Tap, swipe, pinch gestures via libs-gui
- Mobile Lifecycle:
  - Handle onPause(), onResume(), onTerminate()
- Conditional Activation:
  - Mobile code paths are only used on mobile builds
  - Backward-compatible with desktop

VALUE & BENEFITS

- **Unified codebase** for desktop and mobile (Objective-C)
- **Better performance** on iOS due to native integration
- **Broader reach** for apps across more platforms
- Meets modern **cross-platform development needs**
- Attracts new contributors and mobile-focused projects
- Preserves modularity and backward compatibility

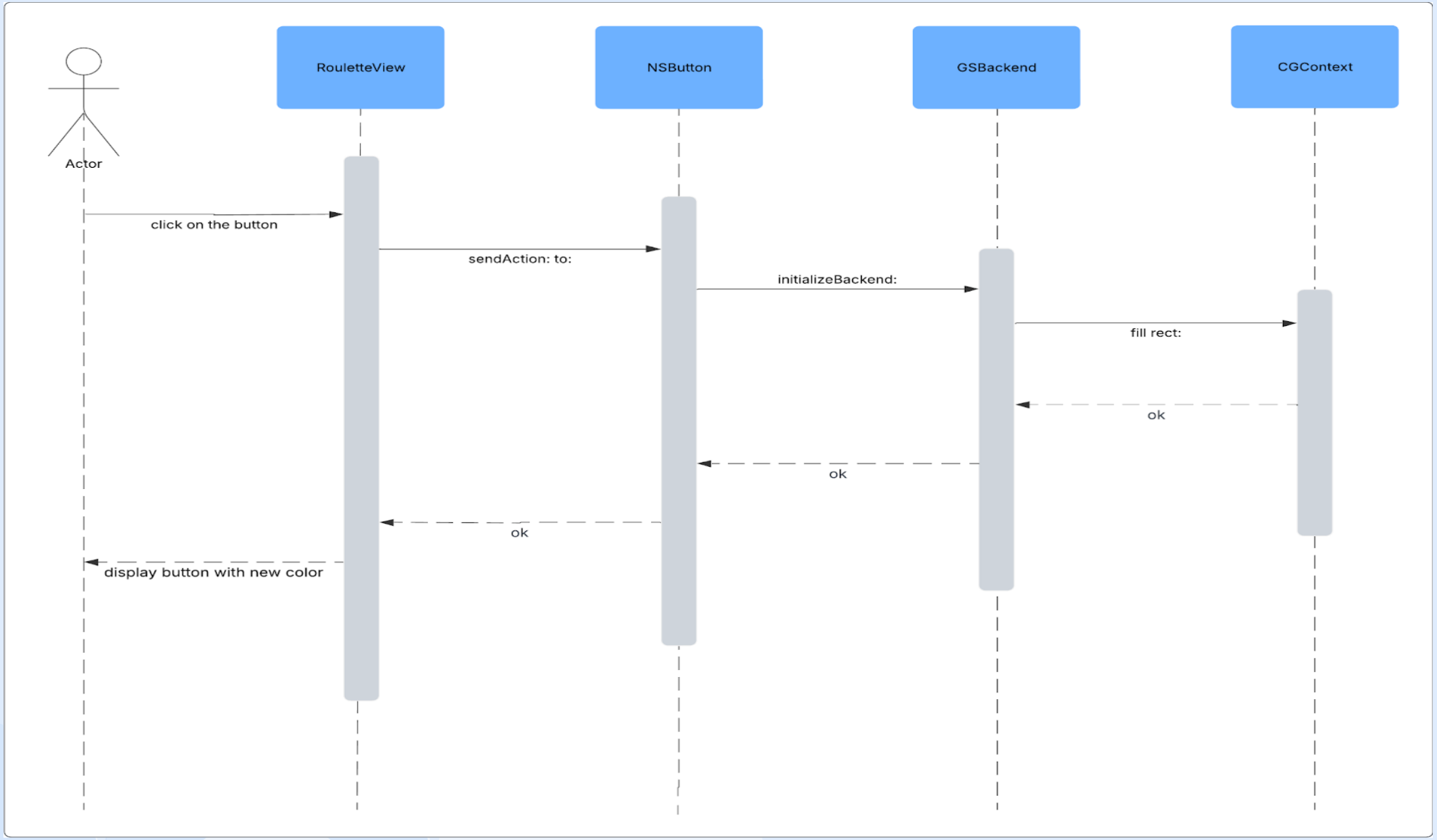
INTERACTIONS WITH EXISTING FEATURES

- libs-back:
  - Adds new rendering backends for mobile platform
  - Integrated via existing rendering interfaces

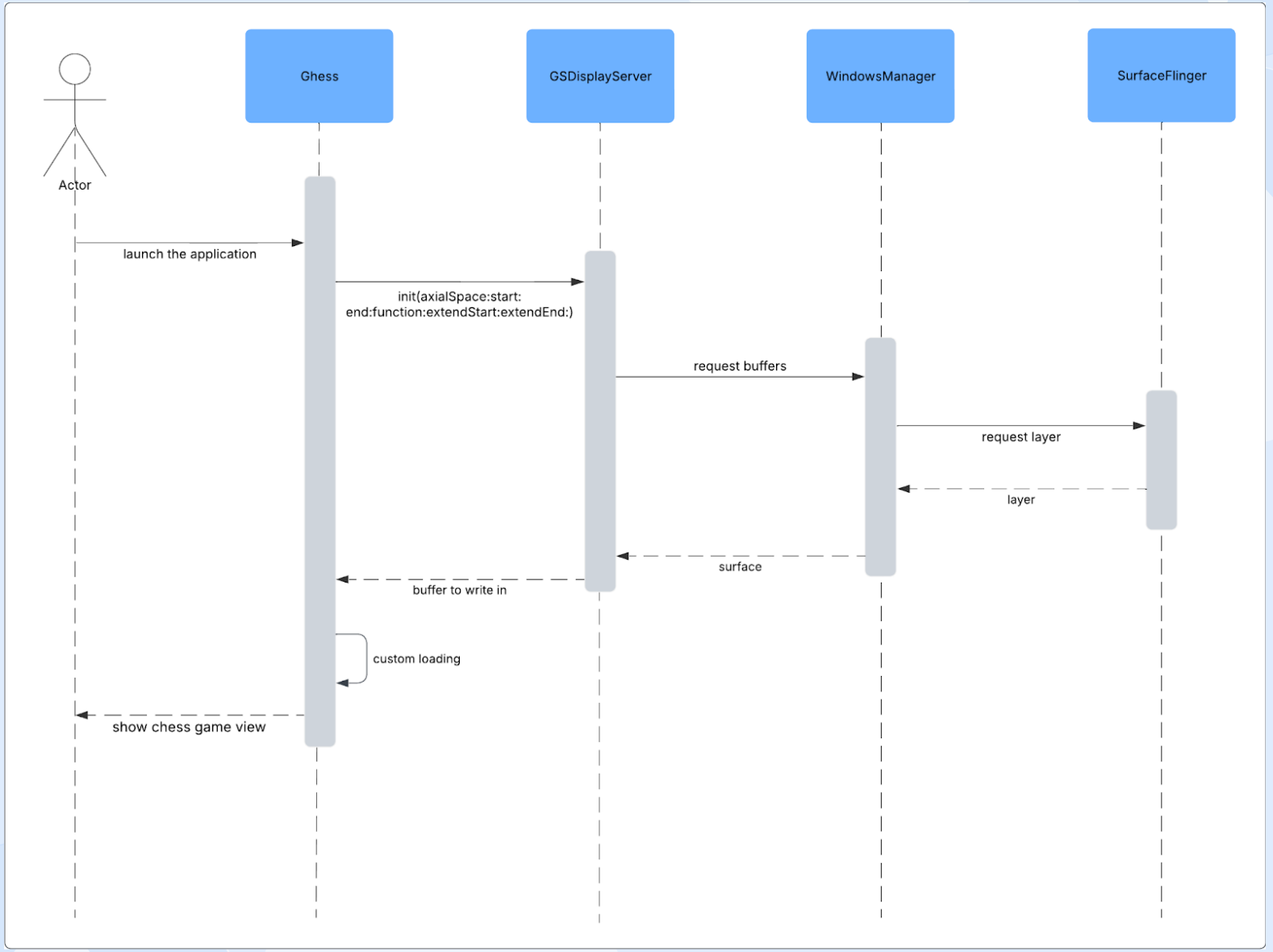
- libs-gui:
  - Introduces touch and gesture handlers
  - Keeps desktop input (mouse/keyboard) unchanged

- libs-base / libs-corebase:
  - Minor updates for Mobile file paths and sandboxing;
  - Device info (e.g., screen resolution, battery)

USE CASE & SEQUENCE DIAGRAM 1



USE CASE & SEQUENCE DIAGRAM 2



SAAM ANALYSIS - STAKEHOLDERS

- Developers
- Developers of GNUStep Libraries
- Maintainers of (GNUStep)
- Mobile Users
- Publishing Platforms (Play Store and AppStore)

SAAM ANALYSIS- NFR

- For maintainers: Maintainable
- Developers: Good developer experience and relatively easy to use
- Play Store and AppStore: Secure and stable
- Users: be efficient on performance, memory, and power usage on mobile devices

EVALUATION OF ALTERNATIVES- REACT NATIVE'S APPROACH

- React Native is another cross-platform mobile framework, using JavaScript
- React components render its JavaScript components in corresponding Native UI Widget
- React Native has a bridge between JavaScript and Native code to communicate between each other, which comes at some compromise of performance

EVALUATION OF ALTERNATIVES - FLUTTER

- Flutter has its own rendering engine, formerly Skia but has been replaced with Impeller
- Its custom rendering engine doesn't use corresponding native UI Widgets and ensures consistency across all mobile platforms
- Flutter uses Dart which is compiled ahead-of-time into native ARM code and thus more performant than React Native Applications
- Because Flutter has its own rendering engine, its applications take up more disk space

SAAM – SELECTED ALTERNATIVE

- Creating an entirely new rendering engine may not be feasible given the resources of the maintainers of GNUStep

- React Native's approach of using React Components in JavaScript to correspond to native UI Widgets is more similar to that of GNUStep, which uses the Cocoa API to map UI elements to native graphics systems of its existing supported UI

- On iOS, GNUStep can have a significant advantage since Objective-C and more recently Swift is the native programming language on the iOS platform

- The risk of this approach is that such complex and comprehensive features, and potential changes to existing subsystems could add complexity that exposes bugs and security vulnerabilities

- Limitations of this approach would be incorporating compatibility between an Objective C written application to Android, which Android doesn't natively support like Java and Kotlin

TESTING

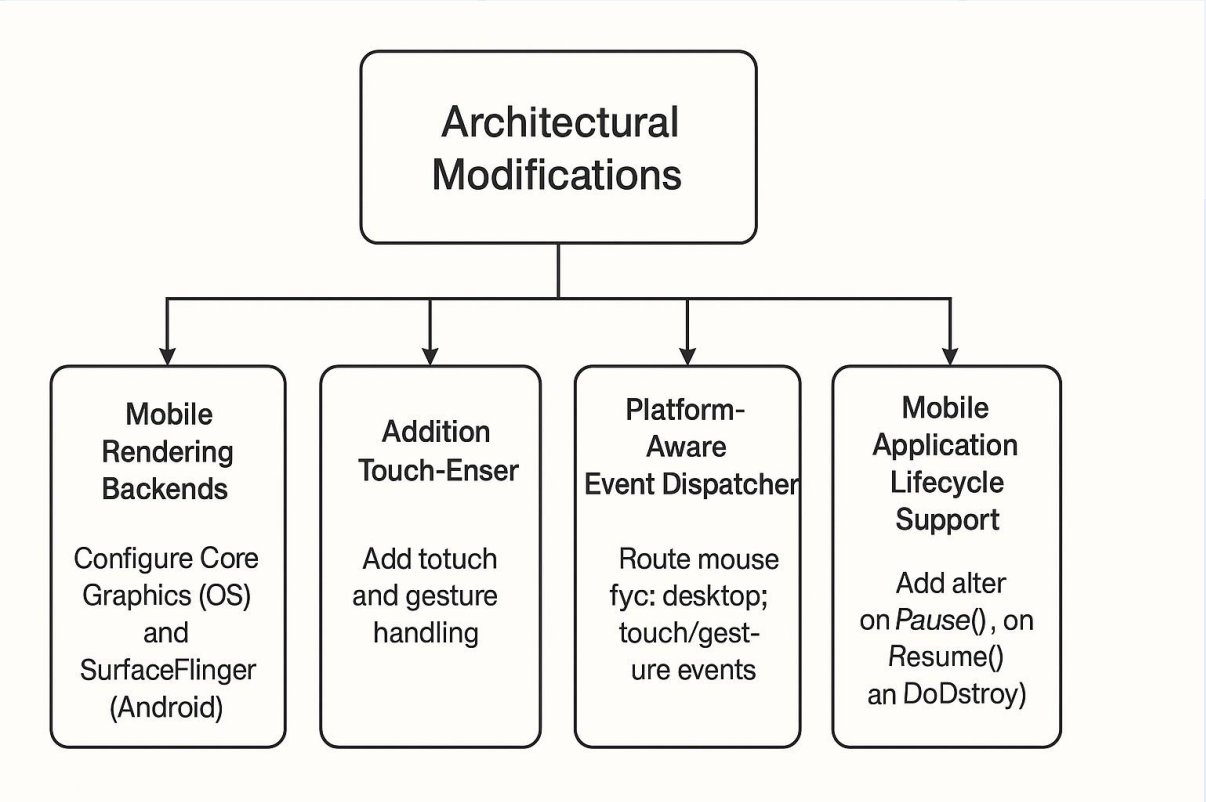
- Developers can test their code on virtual machine providers like Android Virtual Device, or Xcode

- GNUStep can also implement a testing library similar to that of Jest for React Native which tests behaviour of functions and components of the application

- Performance counters can be introduced into the GNUstep project to better monitor that what piece of code will be the bottleneck of the whole program so that it can be optimized

ARCHITECTURAL MODIFICATIONS

- Several new subsystems will have to be implemented to support new features while preserving support for existing OS
- Libs-gui will also need to be updated to support mobile to support touch interactions



LESSONS LEARNED

- We learned that cross-mobile frameworks are complex, mobile applications are more constrained on computing resources in size, power, and processing that don't face the same restrictions as a desktop or laptop may have

- Implementing mobile support isn't just about adding new code, but it affects existing code with regards to rendering, input handling, app lifecycle, and more.

- The best solution isn't always the most powerful one, sometimes, we need to be more pragmatic about the currently available resources of the current team on what they can build and maintain

The End