Enhancement Proposal Report for GNUStep

Group 3: Gamers Never Give Up

Zoya Zarei-Joorshari — 19zzj@queensu.ca

Meagan Mann — 20mmm27@queensu.ca

Jawad Ahmed — 19jaa14@queensu.ca

Kim Hyun Bin — 24vln2@queensu.ca

Ripley Visentin — 22rknv@queensu.ca

Ethan Nguyen — 20hen@queensu.ca

April 5, 2025

Contents

Abstract

This report aims to propose a new feature that will enhance the user experience of GNUStep: Gorm Cloud. This feature will allow users of GNUStep to save their templates, pallettes, or GUIs to the cloud, either publicly or privately. It also allows users to import other user's publicly uploaded templates/pallettes/GUIs into their own project. This report will also perform SEI SAAM analysis to find the appropriate architecture to implement this new feature. Further, this report will discuss the effect of this feature on existing subsystems, plans for testing, and potential risks.

1 Introduction

GNUStep is an open source framework created for GUI development. Written in Objective-C, it is a free, object-oriented, cross-platform, development environment. It inherits from NeXTSTEP-like applications and was designed in mind for cross-compatibility between legacy and modern Apple environments. GNUStep has many key components that have their own set of specific responsibilities.

Our team previously wrote two reports analyzing the conceptual and concrete architecture of GNUStep, noting its layered architectural style with some object-oriented properties throughout the system. In this report, we propose an enhancement to the existing GNUStep system architecture by adding an on-demand cloud service for the Gorm subsystem. This will serve to allow users to develop GUI applications and objects through the internet in order to facilitate cross-collaboration between developers, scalability and ease-of-use. With this, organizations can develop custom-made pallettes for internal use, and enables organizations to store, access, and maintain their applications.

In this report, we illustrate the specifications of the proposed enhancement, how it will impact the components involved, and how the architecture will be affected and why our team made the decision to move forth with this feature. Furthermore we describe the state of the system before and after the enhancement. This will be supported by a robust SEI SAAM Analysis highlighting stakeholders, non-functional requirements and the details of implementation. The next section will describe the effects on subsystems within GNUStep both at a high-level and low-level followed by an examination of testing and potential risks associated with the proposed enhancement. Subsequently, we put forth two use cases for the new cloud enhancement along with associated sequence diagrams to provide clear and concise visualizations for how Gorm-Cloud will be used. Finally, we highlight lessons learned from our team and summarize our findings.

2 Proposed Feature

2.1 Gorm Cloud

After discussion and consideration, the team came up with a new feature for GNUStep: this feature would be dubbed "Gorm Cloud". Gorm Cloud would allow the user the option to upload their templates, pallettes, and GUIs to the cloud. This upload can be

public, visible to other users, or private, only visible to the user who uploaded. Further, other users can browse publicly uploaded templates/pallettes/GUIs and import those that interest them into their own projects.

This feature was chosen as it would provide a major enhancement to the GNUStep user experience. Saving templates/pallettes/GUIs locally can be expensive for the user in terms of storage: Gorm Cloud would alleviate this issue as it would allow the user to store their templates/pallettes/GUIs on the cloud. Further, this feature will simplify GUI development for users, as it allows them to more easily gain access to references GUIs and custom templates/pallettes through being able to browse other user's uploads and import them into their own projects. As a result of the above enhancements, Gorm Cloud will allow for a more accessible GUI development experience for all users.

2.2 Interaction with Other Features

With this enhancement, there will not be much of a difference in the standard ways of how GNUStep is used. The main change that would be present within the Gorm app is that it would need to interface with a server so that users may upload their templates/pallettes/GUIs to the cloud. All other features of GNUStep should remain unchanged.

3 Current State of the System

The current state of the system is as shown in Figure 1: this architecture was derived in the previous Concrete Architecture Report. In the previous report, the layered architecture was chosen as it improved dependency management, modularity, and maintainability of the code. In this report, we will modify this architecture to implement cloud capabilities for GNUStep.

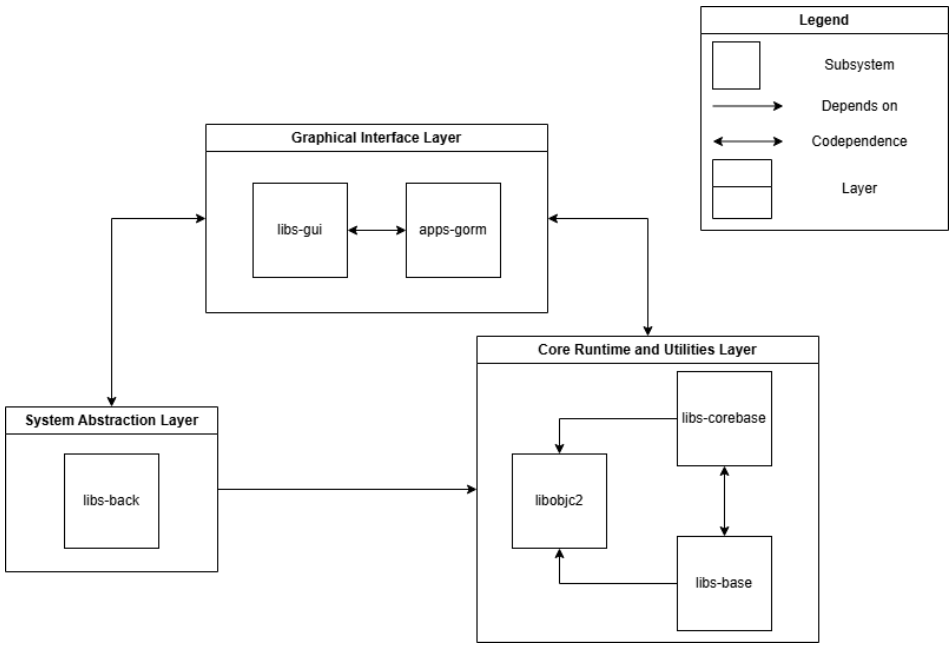

Figure 1: Current Architecture of GNUStep

4 SEI SAAM Analysis

4.1 Identifying Stakeholders

As per our SEI SAAM architectural analysis, we identify major stakeholders of the proposed enhancement to GNUStep. First and most important are the end-users, the developers who use GNUStep and those who have made open-source contributions to the system. Ultimately, any proposed enhancements are for them first and foremost as they are the ones who provide GNUStep with it's engagement and functionality. On the same note, our project maintainers and contributors will also be considered with the same degree of attention. More specifically, this proposed enhancement will be beneficial for larger organization development teams, especially those with remote/hybrid work policies across time zones. The addition of a cloud service on the Internet greatly facilitates collaboration and effectiveness for these developers seeking to develop apps without being limited by their local machine.

Another major stakeholder would be cloud infrastructure providers. Services like AWS and Azure provide hosting, scalability, and infrastructure to cloud platforms and serve as an important reference and potential partner for endeavors. Any investors and funding bodies will also be considered as important stakeholders as they provide the capital for GNUStep to improve its functionality. Finally, our own development team, from QA to project managers are identified as major stakeholders. As they build the proposed enhancement, their voices will be considered to great effect as they know the whole system architecture from each level of abstraction.

4.2 Identifying Non-Functional Requirements

With these stakeholders identified, we now define the key non-functional requirements (NFRs) that will ensure the proposed GNUStep enhancement meets their needs effectively.

End-Users

1. Usability: The cloud-enhanced GNUStep must have an intuitive and developer-friendly UI/UX.
2. Performance: The system should provide low-latency access to cloud-based development resources.
3. Security: Must ensure secure authentication and data encryption.

Project Maintainers and Contributors

1. Maintainability: The system should have a well-documented modular code for easy maintenance.
2. Scalability: Must support an increasing number of contributors without performance degradation.

Larger Organization Development Teams

1. Collaboration: The cloud service should enable real-time collaborative development.
2. Cross-Platform Support: Must function consistently across macOS, Windows, and Linux.

Cloud Infrastructure Providers

1. Interoperability: The enhancement should support deployment on multiple cloud platforms.
2. Security and Compliance: Must adhere to industry security standards.

Investors and Funding Bodies

1. Cost Efficiency: The solution should be cost-effective with a clear ROI potential.
2. Market Competitiveness: The system should offer unique advantages over existing solutions.

Development Team

1. Testability: The system should include automated testing capabilities.
2. DevOps Integration: Must support CI/CD pipelines for efficient deployment.

4.3 Implementation Methods

4.3.1 Implementation 1

In this implementation, the apps-gorm subsystem would still exist on the client-side while a new subsystem would be added as the server: gorm-cloud. The gorm-cloud subsystem would include a database to store uploaded templates/pallettes/GUIs, it would also manage data received and requests for data retrieval from apps-gorm. The apps-gorm subsystem would need an added component that would facilitate communication with the gorm-cloud subsystem: CloudInterface. The addition of the CloudInterface component would add a dependency from apps-gorm to gorm-cloud: CloudInterface will be the connection from apps-gorm to gorm-cloud, it will also call on gorm-cloud when data needs to be uploaded or retrieved. All other apps-gorm subsystem dependencies would be unchanged: data retrieved from gorm-cloud would be treated the same as data stored locally and rendered in the same manner.

From this implementation description, it can be seen that a client-server architecture would best suit this feature, as it allows for simple distribution of data, and allows for easy scaling of the server if the user base increases. This implementation would also impose minimal changes to the original architecture as it allows the client-side to continue using the original layered architecture while only adding a single component on the server-side. Figure 2 displays the updated architecture using this implementation with the new subsystem and dependency shown in red.

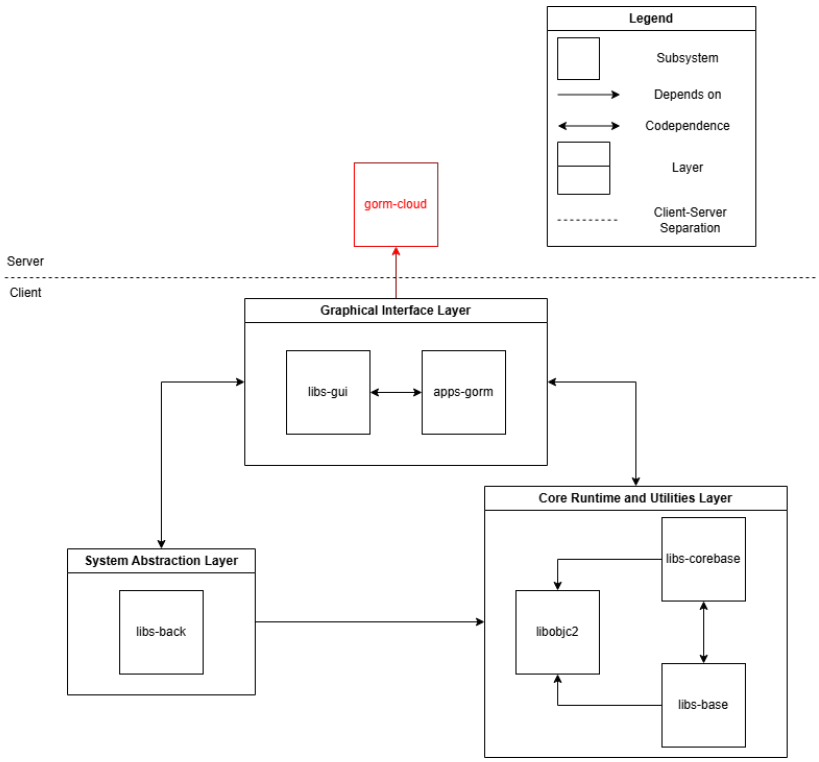

Figure 2: Updated Architecture Using Implementation 1

4.3.2 Implementation 2

Similar to the initial implementation, the second implementation alters the structure of GNUStep through integrating gorm into a cloud-based environment within the client-side application into the apps-gorm subsystem, which includes other modules within the server-side. Instead, this approach differs through its introduction of an advanced cloud manager and its various microservices within apps-gorm. These enhancements would function to manage a variety of responsibilities, including local data storage, caching, and asynchronous synchronization. By interpreting the modular structure of this implementation in figure 3, we can observe that each of the components within the client-side of the apps-gorm subsystem is capable of being independently scaled, which yields an improved scalability while reducing overall latency in accessing the cloud.

There are many approaches to enhancing GNUStep's architecture, yet this implementation differentiates itself from the others by its emphasis on increased modularity and scalability, optimizing its performance to allow users to efficiently access and manage data stored onto the cloud with ease.

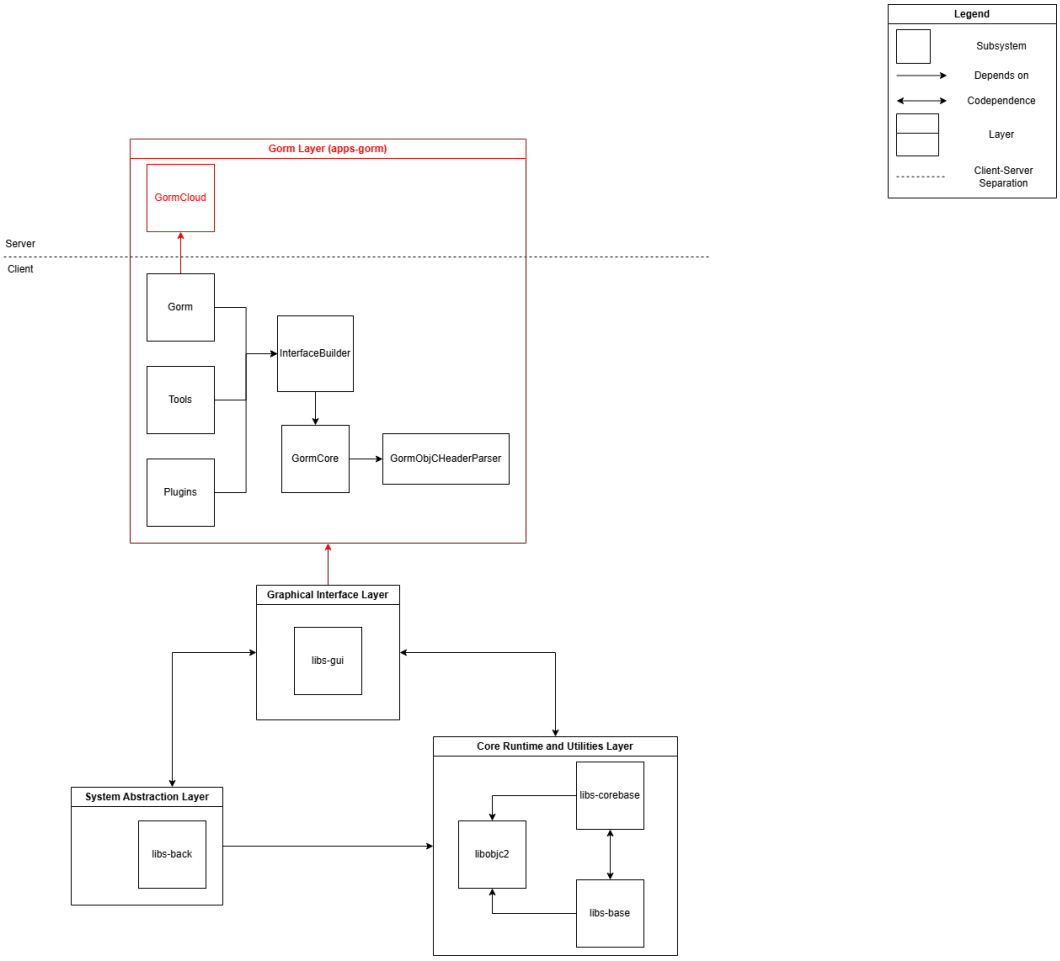

Figure 3: Updated Architecture Using Implementation 2

4.4 Comparison of Implementations

Feature	Client-Server Implementation	Modular Cloud Implementation
Architecture	Client-server model with a new gorm-cloud subsystem. Minimal changes to existing architecture	Cloud-integrated apps-gorm with microservices. Modular architecture with more integration on client side.
Data Storage	Centralized database in gorm-cloud. Users can add local assets to the centralized database.	Hybrid storage system. Distributed storage with local caching.
Communication	Uses CloudInterface for data retrieval. Simple communication using direct HTTP requests.	Asynchronous synchronization for efficiency, changes synchronized in the background.
Scalability	Server can be scaled separately with more resources for the centralized database.	Modular structure allows for independent scaling. Scaling can also occur on the client's side with the use of local storage.
Performance	Simple but may introduce latency. Performance may vary depending on network speed, geographical location, and cloud infrastructure.	Optimized for low latency and fast access. Asynchronism with caching allows for faster handling of requests. Use of microservices distributes load more efficiently.

5 Effects on Subsystems and Directories

Building upon the proposed 'Gorm Cloud' enhancement, the following sections explore its impact on various subsystems, from high-level architectural changes to low-level subsystems and the necessary adjustments to directories and files.

5.1 Effects on High-Level Subsystems

With the introduction of the Gorm Cloud feature, the high-level subsystems within GNUStep will need to evolve to incorporate cloud-based interactions. The biggest change will be the introduction of the gorm-cloud subsystem, which will manage cloud storage and interaction with the apps-gorm subsystem. This will directly affect the systems overall architecture by introducing the client-server model. The apps-gorm subsystem remains on the client side, while the gorm cloud subsystem will manage cloud storage and data retrieval. Key changes include the added CloudInterface component that communicates with the server side gorm-cloud subsystem and the gorm-cloud subsystem will require its own data base to store uploaded templates, palettes, and GUIs.

5.2 Effects on Low-Level Subsystems

At the low-level the CloudInterface component will be introduced in the apps-gorm subsystem. This component will handle the communication between the client-side and the

cloud, ensuring smooth data transfer for templates, palettes, and GUIs. Additionally, the interaction between gorm-cloud and apps-gorm will introduce a new dependency. This will need to be carefully managed to avoid performance bottlenecks or unintended disruptions. Once the data is retrieved from the cloud it must be processed and rendered in the same way as the local data. This means that the existing data handling mechanisms in apps-gorm will not change significantly.

5.3 Effects on Directories and Files

The enhancement will necessitate changes in the directories and files structure. Specifically, the addition of gorm-cloud will require new directories to manage server-side functionalities related to cloud storage. In the apps-gorm subsystem, files related to CloudInterface will be introduced to handle the communication between the client and the server. They will handle interactions with the cloud, such as uploading data to the server and retrieving when necessary. Furthermore, the system will need to be reorganized to handle the separation of cloud stored templates, palettes, and GUIs, as well as the management of public and private access controls for user uploads.

6 Potential Risks

Conclusively, the implementation of a cloud-based system to GNUStep's gorm component grants us several benefits, as mentioned before, such as scalability and efficient data accessibility. However, it is inevitable that the introduction of potential enhancements and solutions come with their own problems and risks. The employment of gorm-cloud GNUStep increases the scope and complexity of the software architecture with its addition of new subsystems and/or microservices, which introduces complications to its development and maintenance. The failure of each component in part can easily disrupt the entire system. Systems dependent on cloud interactions also depend on proper network connections, leaving the system exposed to potential latency and bottleneck issues. Because of these risks, it is important to understand the development of gorm-cloud without overlooking the potential issues that may arise.

7 Testing

Testing is essential in ensuring that our new Gorm Cloud feature can be seamlessly integrated into GNUStep. The following testing plan assures that the feature works as intended, meets user expectations, and does not introduce new issues into the system.

7.1 Basic Unit Testing

Test the new CloudInterface component, ensuring that it effectively bridges the gap between gorm-cloud and apps-gorm. CloudInterface should be able to handle uploads and downloads to and from the cloud without any data issues or bugs.

7.2 Client-Server Communication Testing

Various user tasks such as browsing public templates or uploading templates to the cloud should be simulated to check for proper synchronization. Errors, such as network failure or data corruption, should also be simulated to examine the behaviour of the application in these scenarios.

7.3 Performance testing

The latency of the application for upload / download operations should be tested to ensure that it is within a specified limit. Load testing should also be performed to assess performance in a post-deployment environment, where many users may be uploading/downloading at the same time. This can be done with tools such as Apache JMeter and Pylot. Finally, scalability can be tested through examining the application's performance under various stress levels. This will allow for identification and plans for alleviation of performance degradation.

7.4 Security Testing

Due to the introduction of cloud services, there is a new risk to user data and the application must undergo corresponding testing to minimize this risk. Processes such as user authentication and data encryption should be tested for adequacy.

7.5 User Experience Testing

The final stage of testing involves the ease of usability for the average user. The UI for the various functions of the application, such as uploading/downloading and asset browsing, should be intuitive and developer-friendly.

8 Sequence Diagrams

8.1 Use Case 1: User Saves a Custom GUI to the Cloud Publicly

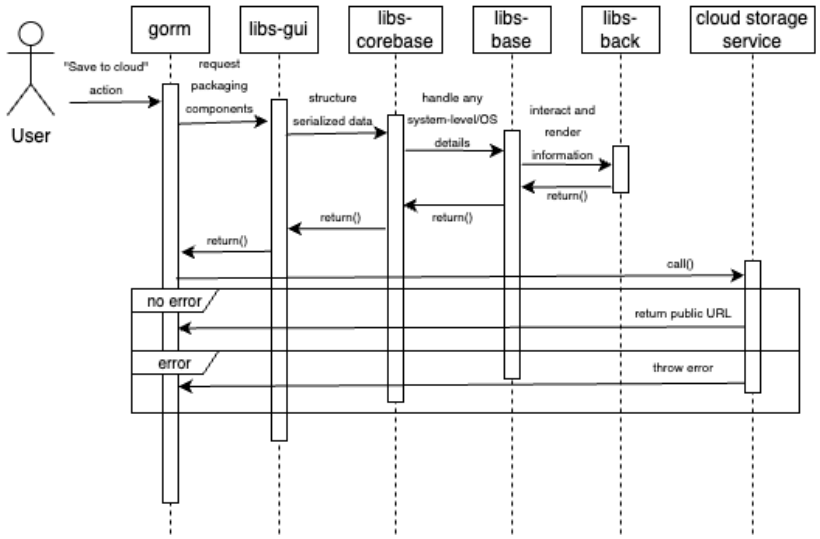

Figure 4: Sequence diagram for Use Case 1

8.2 Use Case 2: User Downloads a Custom Template from the Cloud

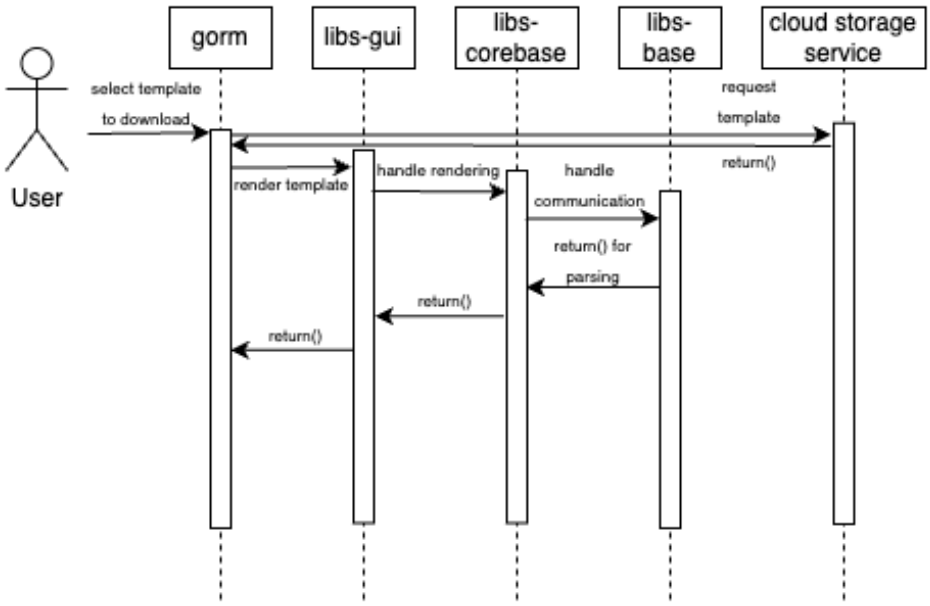

Figure 5: Sequence diagram for Use Case 2

9 Dictionary

UI/UX - User Interface / User Experience
CI/CD - Continuous Integration / Continuous Deployment
ROI - Return on Investment
NFR - Non-Functional Requirement
GUI - Graphical User Interface
UI - User Interface
Cloud - Online storage space to store files and applications [1]
Server - Computer or system that provides information to other computers[2]

10 Lessons Learned

Completing our research into GNUStep's weaknesses and proposing the Gorm Cloud system as an enhancement revealed the benefits in its implementation, such as scalability, centralized data management, and improved accessibility. However, employing such a change would bring about various issues, such as latency, security over the network, synchronization problems, and increased structural complexity. Although the completion of the enhancement proposal was the final objective, the process of brainstorming and constructing our report rewarded us with a stronger understanding of the structure of GNUStep, especially the strengths and flaws of the software.

11 Conclusion

This report proposed a new feature for GNUStep: Gorm Cloud. This feature will allow for a more accessible GUI development experience for all user's. There are several advantages to incorporating a cloud-based component to GNUStep. These include but are not limited to: enabling scalability for large scale and long term development through a centralized cloud storage system, faciliating collaboration and accessibility across multiple environments and development teams, and modernizing GNUStep's architecture to align closer with current software development practices. The implementation and updated architecture for the feature was derived using SEI SAAM analysis, illustrating major stakeholders, non-functional requirements and implementation details. While the layered architecture was discussed in the previous Concrete Architecture report, the client server architecture was chosen to accommodate this new feature.

Further effects on subsystems and directories, potential risks and testing were discussed to mitigate potential issues in development and illustrate the planning required to incorporate the proposed enhancement to a large system architecture with varying levels of component interactions. This is supported by our presented use cases with associated sequence diagrams to provide a holistic view on the potentials for the proposed feature.

Ultimately, Gorm-Cloud represents a significant step forward in modernizing the GNUStep ecosystem by introducing cloud-based functionality to the existing Gorm component. Although it introduces added complexity and new architectural considerations, the

benefits of improved user experience, flexibility, and future proofing of GNUStep's development tools make Gorm Cloud a valuable and necessary enhancement in the ever-changing software development landscape.

References

[1] Microsoft. What is the cloud?

[2] Wikipedia contributors. Server (computing).