

**Preface**

Welcome to the FluentCraft Framework design document. This document will provide a detailed overview of the architecture, components, and design considerations of our new Qt-based UI component library that is mirroring aesthetically and functionally Microsoft's Fluent UI.

**Acknowledgments**

I extend my heartfelt thanks to *Dr Amin Amini* my project supervisor, for his invaluable guidance and expert advice throughout the development of the FluentCraft project. His contributions were essential to the success of this work.

I am also grateful to the Computer Science Department at UCLan for providing the necessary resources and technical support required to complete my project, especially the lab technicians in the software development lab.

Finally, I would like to thank my family for their constant encouragement and support during my studies, which has been a cornerstone of my success.

**Contact Information**

***Organization***

University of central Lancashire.

Telephone 01772 201201

Campus: Preston, England.

More Info at: www.uclan.ac.uk

***Author,***

Name: Jacob Masih

Website: JacobSidhu.com

Course: Software Engineering.

GitHub: @jacobsidhu

1. **Introduction**
   1. Purpose of this document.
   2. Project Identification.
   3. Scope of the Framework.
   4. Relationship to Other Software Components.
   5. Development Methodology, Tools, and Techniques.
   6. Compliance with Standards and Regulations.
2. **Framework Overview**
   1. Background and Motivation.
   2. Current State of QT UI Components.
   3. Proposed Enhancements in FluentCraft.
   4. Long-Term Vision and Technology Forecast.
   5. Constraints and challenges.
   6. Design Trade-offs and Decisions.
   7. Target Audience and user characteristics.
3. **System Architecture**
   1. Overall Software Architecture.
   2. Component Hierarchy and Interactions.
   3. Themes and Customization Mechanisms.
   4. Integration with Existing Qt Environments.
   5. Data Flow and Management.
4. **Component Design**
   1. Catalogue of Core Components.
   2. Component Specifications.
   3. Component States and Behaviours.
   4. Data Binding and Event Handling.
5. **Interface Design**
   1. API Design Principles.
   2. Interface Documentation Standards.
   3. External Interface Design.
   4. Accessibility Features.
6. **Testing and Validation**
   1. Testing Strategies and Frameworks.
7. **Deployment and Maintenance**
   1. Deployment Strategy.
   2. Update and Upgrade Mechanisms.
   3. Support and Troubleshooting.
8. **Appendices**
   1. Requirements Traceability Matrix.
   2. References and Resources.

**TABLE OF CONTENT**

**1 Introduction**

**1.1 Purpose of the document:**

This document describes in detail the architecture and design of the FluentCraft framework, concretely stating what it is intended for, how it has been designed, and how the different components interact with each other. The present document will be useful to developers, give the guideline and blueprint to project managers, and represent the proposal for the stakeholders in order to align and make sure that all parties understand what is proposed.

**1.2 Project Identification**

* Project Name: FluentCraft
* Project Manager: Jacob M.
* Project Code: ------
* Project Organiser: University of central lancashire.

**1.3 Scope of the Framework**

The FluentCraft framework is designed to encapsulate the essence of Microsoft's FluentUI 2 design system and offers to developers a powerful toolkit aimed at accelerating their application development with Qt. Development will be versioned, each version developed and released to make sure that a whole, polished, and robust set of UI components for different application needs is delivered.

In the FluentCraft framework, we want to support your different needs in modern software development with a complete set of UI components. Below is a detailed description of all components planned for the future release of this framework. Usability, accessibility, and aesthetic consistency are the guidelines in their design so that they not only reach the highest functionality level but also perfectly fit into existing Qt applications.

This is a forward-looking roadmap for components that enables us to further develop and enhance the FluentCraft framework as a must-have tool for any developer willing to deploy sophisticated, modern user interfaces.

|  |  |
| --- | --- |
| PROJECT VISION | |
| Animated icon | Menu flyout |
| Animated Visual Player | Navigation view |
| Auto-suggest box | Number box |
| Breadcrumb | Parallax view |
| Button | Password box |
| Calendar date picker | Person picture |
| Calender view | Pips pager |
| Check box | Progress bar |
| Color picker | Progress ring |
| Combo box | Radio button |
| Command bar | Rating control |
| Command bar flyout | Repeat button |
| Content card | Rich edit box |
| Content dialog | Rick text block |
| Content link | Scroll viewer |
| Context menu | Semantic zoom |
| Date picker | Shapes |
| Dialogs and flyouts | Slider |
| Drop down button | Split button |
| Expander | Split view |
| Flip view | Swipe control |
| Flyout | Tab view |
| Forms | Teaching tip |
| Grid view | Text block |
| Hyperlink | Text box |
| Hyperlink Button | Time picker |
| Images and images brushes | Toggle switch |
| Info bar | Toggle button |
| lnking controls | Toggle spit button |
| List/details | Tooltip |
| List view | Tree view |
| Map Control | Two-pane view |
| Media Playback | Web view |
| Manu bar |  |

Version v\_1.0.0 will be about the integration of the Fluent UI design system, hence making this a Minimal Viable Product version. Basically, it will provide the basic controls any application interface needs: buttons, windows, dialog boxes, and input fields to let the developer construct the most basic yet functional interfaces efficiently.

Below is what a developer can expect from version v\_1.0.0 in tabular form to give a proper roadmap of the components included, stating the quality and usability a framework promises to give to its users:

|  |
| --- |
| **Included** *v\_1,0.0* |
| |  |  | | --- | --- | | Basic Input | | | 1. Button |  | | 1. HyperlinkButton | | | 1. ToggleButton | | | 1. Checkbutton | | | 1. ComboBox | | | 1. Slider | | | 1. RadioButton | | | 1. ToggleButton | | | 1. RepeatButton | | |
| |  | | --- | | Text | | 1. TextBox | | 1. TextBlock | | 1. PasswordBox | |
| |  | | --- | | Status & info | | 1. InfoBar | | 1. ProgressBar | | 1. ProgressRing | |

Looking Ahead to Future Versions

As we continue to refine and expand FluentCraft, the next version will aim to incorporate additional features based on user feedback and emerging technology trends. Future updates are planned to include more customizable components, improved accessibility features, and enhanced integration capabilities with other software tools. These enhancements will further our commitment to delivering a versatile and user-friendly UI component library that meets the evolving needs of developers and end-users alike.

**1.4 Relationship with other software Components**

The FluentCraft framework is designed to be integrated into an existing Qt application. The main points of interaction will be:

* Qt Framework: FluentCraft extends the Qt Framework by introducing additional UI elements composed of Fluent Design System principles, and it should be fully compatible with Qt 5.x and Qt 6.x software in order to go along with the current Qt applications.
* Dependency: FluentCraft directly depends on Qt Core and Qt GUI for minimal functionality. It uses the QML module for designing an interactive user interface.
* Common Components: FluentCraft shares the QtQuick module with other systems in the development of a fluid, animated user interface. Changes to QtQuick shall not make it incompatible with FluentCraft.
* Compatibility/Co-existence: FluentCraft has been designed to co-exist with other third-party Qt extensions. Care has been taken to reduce conflict with some of the commonly used Qt libraries like KDE frameworks and Qt Extended.
* Security Implication: Interactions with the external components shall be done via secure QML interfaces, and any data exchange is performed using the SSL / TLS protocols to keep the integrity and confidentiality.

Future Upgrades:

FluentCraft will support future Qt Frameworks through modular design and by paying attention to Qt's forward compatibility guidelines.

**1.5 Development Methodology, Tools, and Techniques.**

It is designed to use agile project management practices and solid technical tools to achieve a smooth and efficient development process while developing the FluentCraft framework.

Methodology:

* Scrum: We will be adapting the Scrum framework in managing the development of FluentCraft. The framework follows an iterative and incremental approach. It keeps our team agile and responsive to changes in the needs of the stakeholders. We enable our team to constantly assess the deliverables and priorities through regular sprints: Daily stand-ups, sprint reviews, and retrospectives will be part of our process to keep everybody informed and allow time for readjustments of our project plan.

Tools:

* CMake: This is a build system tool used in automating the compilation, testing, and packaging of the FluentCraft framework. It ensures our build process is platform-independent and maintainable on any operating system, thus making the integration and deployment of our product much easier.
* Figma: It is used for high-fidelity prototyping and allows our design team to visualize interfaces and experience flows before they are implemented, thus supporting early feedback and iterative adjustments.
* Qt Creator: As the fundamental development environment, Qt Creator is feature-rich, providing a rich playground for coding, testing, and debugging the UI components within FluentCraft, thus enhancing productivity and quality.
* Git: As the basic version control system, Git helps in supporting collaborative development on the code; hence, multiple team members can concurrently work on projects without any issues and yet keep the history and integrity of our codebase intact.
* Trello: As the project management tool for managing traceability, Trello provides a visual task management interface that allows tracking the progress made within each sprint, assigning tasks to individuals, and monitoring the deadlines for projects.
* Accessibility Standards: Accessibility standards are followed throughout the development life cycle to guarantee that FluentCraft components are usable by people to the largest extent possible, including people with disabilities.

By combining, FluentCraft project is set to deliver an intuitive, feature-rich UI component library that will not only advance functionality and visual appeal for Qt applications but also make sure that fluent integrations are achieved with ease of maintenance in diverse development environments.

**1.6 Compliance with Standards and Regulations.**

The FluentCraft framework is developed in compliance with many standards and regulations that govern software quality, security, and accessibility. Compliance with these standards makes the product much more usable and safer, while being more marketable, able to meet the expectations and requirements of a wide range of users and regulatory bodies.

Areas of compliance include:

* Software Quality Standards: FluentCraft adheres to the ISO/IEC 25010 standard, which sets up a set of characteristics regarding software product quality. The standard covers functionality, reliability, usability, efficiency, maintainability, and portability, thus guaranteeing stability and dependability for the framework.
* Accessibility Standards: Adherence to the standards WCAG 2.1 AA compliant and ADA compliant ensures that components have been designed to be usable by people with disabilities. That means keyboard navigation, screen reader support, proper visual design in regard to all sorts of color blindness, and other visual impairments.
* Data Protection and Privacy Regulations: FluentCraft is fully compliant with the General Data Protection Regulation for users of the European Union, ensuring whatever information is gathered from using our framework is kept securely and transparently. These include strong data encryption, secure storage of data solutions, and clear policies of privacy regarding the use, storage, and protection of data.
* Security Standards: The FluentCraft solution attaches the highest premium to security. We follow best practices laid down by OWASP to avoid such basic security issues as SQL injection, XSS, etc.
* Internationalization and Localization Compliance: FluentCraft is out-of-the-box compatible with i18n and l10n standards, easily customizable for any language and region with little or no change in the software structure.

Monitoring for Compliance

The FluentCraft team will perform timely checks and audits of the codebase and processes to maintain compliance. We will contribute with experts in multiple respected fields, including law and industry-specific fields, which includes continuing to develop new and evolving standards and regulations affecting our products. In addition, we train our team on such compliance-related issues that they will be well and fully integrated into their everyday development activities.

**Framework**

2 **Overview**

**2.1 Background and Motivation**

The concept of FluentCraft raised is from the ever-growing demand for more modern and user-friendly interfaces among desktop applications. The continuous evolution of standards and expectations, especially under the influence of large platforms like Windows and its Fluent Design System, left a large void that needed to be filled with high-quality, similar UI components for Qt applications. There are many existing Qt applications don't fit the modern aesthetics and intuitive functionalities expected by users who have gotten accustomed to them, hence narrowing their usage and overall usability.

This need inspired FluentCraft to fill in the gap and provide a library of Qt-based components carved out of the philosophy of Microsoft's Fluent UI, which will enable developers to create applications that will be functional, good-looking, and offer a user experience that is elevated to modern design trends and standards of accessibility.

* 1. **Current State of QT UI Components.**

Qt is a mature cross-platform development framework, and it indeed provides a solid foundation of tools and libraries for creating graphical user interfaces. It offers a complete set of UI components through its modules like QtWidgets and QtQuick, which allow both the development of traditional desktop applications and modern touch-enabled ones. Still, even with all these capabilities, there are some points where the current state of Qt UI components could be improved:

* Visual Design: Although Qt provides a broad range of styling and theming possibilities, the default look-and-feel of Qt widgets often cannot compete with the state-of-the-art designs imposed by more recent UI paradigms like Google's Material Design or Microsoft's Fluent Design. That is especially true for applications that require a very modern look-and-feel.
* Consistency Across Platforms: Qt is very good at cross-platform compatibility; still, ensuring visual and functional consistency across operating systems is often hard to achieve. It usually requires significant effort from developers to tune UI components to make sure that applications will look and behave identically across all platforms.
* Advanced Animations and Transitions: Well-designed animations and transitions, which are not only aesthetically pleasing but also help the user better conceptualize how an application flows, are increasingly common in modern interfaces. Some of this is achievable using Qt Quick; however, the amount of detail involved often pushes the limits of most developers' capabilities.
* Accessibility Features: In any given application, accessibility will be very important to address inclusivity. Qt has a basic level of accessibility built into it; however, there is much to be added to handle the UI components for advanced accessibility standards, such as those mandated by governmental regulations.
* Touch and Gesture Support: As touch screens proliferate, the demand has grown louder for intuitive touch- and gesture-based interactions. Qt has done a good job of making touch-based input work with Qt Quick; however, for QWidget-based applications, additional work is often necessary to properly handle these new interactions.

**2.3 Proposed Enhancements in FluentCraft**

FluentCraft is going to be a major revamp for Qt-based applications because it introduces a new set of UI components that conform fully to up-to-date design principles, just like those applied in Microsoft's Fluent Design System: everything from making the interfaces beautiful and advanced animation supported, to comprehensive touch and gesture support for enhanced beautification and interactiveness. There is also a lot of attention to accessibility; therefore, each component will have deeper integration of support for the most important features that can be fine-tuned to various users' needs. Finally, performance optimizations are present, designed to make efficient use of system resources without compromising application speed or responsiveness. All of these come with extensive documentation and developer tools that provide for easy integration and customizing, along with community and professional services that back this technology, ensuring that its adoption would similarly be painless in both Open Source and Enterprise environments.

**2.4 Long-Term Vision and Technology Forecast**

In the long run, FluentCraft is envisioned to be the developer's first choice for building state-of-the-art, beautiful, and very functional user interfaces with Qt. It is in our vision that FluentCraft will keep evolving to follow, anticipate, and make good use of the emerging trends in both software development and design philosophy.

**2.5 Constraints and Challenges**

In developing the FluentCraft framework, a good number of constraints and challenges have to be worked out that may influence the scope and delivery of the project. Some of the major issues are that the resources will be limited, as the project relies very much on community contributions and probably variable funding. Technological compatibility with various Qt versions and platforms is one of the biggest challenges, and one has to balance between supporting older systems and providing the latest features. Similarly, performance optimization with rich features, particularly on low-end devices, needs balancing. Stringent incorporation of comprehensive accessibility features to conform to international standards takes more time and expertise, probably slowing down development.

**2.6 Design Trade-offs and Decisions**

These, among other challenges, have necessitated a number of strategic trade-offs and decisions. Thus, the framework points to core platforms in relation to resources utilization for maximal effect, an aspect which will continuously expand with the project expansion. The modular design facilitates the inclusion by end users of only what they need in their systems. Feature development in general will focus on the components expected to bring about greatest improvements in usability, with the deferral of less critical features to later in the development process. Some resource constraints have continuously been relieved from leveraging the development and bug-fixing of the features in this set by the open-source community.

**2.7 Target Audience and User Characteristics**

The targeted users of the FluentCraft framework are software developers, UI/UX designers, small to medium enterprises, and educational institutions. In regard to developers, it will target those currently using Qt who want to extend their applications with modern, attractive UI components. The UI/UX designers will see value in this project when they need tools capable of turning modern design into functional application components. To SMEs, FluentCraft offers a cost-effective means to rapid development of professional-grade, cross-platform applications. Of special importance will this framework be to educational institutions and students interested in mastering state-of-the-art UI development practices.