<Chat Away>

# Software Design Template

Version 1.2

## Table of Contents

[Software Design Template 1](#_Toc106744163)

[Table of Contents 2](#_Toc106744164)

[Document Revision History 2](#_Toc106744165)

[Executive Summary 3](#_Toc106744166)

[Requirements 3](#_Toc106744167)

[Design Constraints 3](#_Toc106744168)

[Rationale 3](#_Toc106744169)

[Recommendations: 3](#_Toc106744170)

[Operating Platform: 3](#_Toc106744171)

[Operating Systems Architectures: 4](#_Toc106744172)

[Storage Management: 4](#_Toc106744173)

[Memory Management: 4](#_Toc106744174)

[Distributed Systems and Networks: 4](#_Toc106744175)

[Security: 4](#_Toc106744176)

[Evaluation 4](#_Toc106744177)

[Server Side 4](#_Toc106744178)

[Client Side 5](#_Toc106744179)

[Development tools 5](#_Toc106744180)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/10/2022 | Jacob Hershberger | Initial Design and requirements |
| 1.2 | 06/19/2022 | Jacob Hershberger | Project Three Requirements |
| 1.4 | 06/21/2022 | Jacob Hershberger | Project Two Requirements |

Instructions: Fill in all bracketed information on page one (the cover page), in the Document Revision History table, in the footer, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## Executive Summary

*Wanting to expand from their existing website, Chat Away is wanting a mobile app developed with a goal of increasing their customer base. In addition to increasing their base, they wanted to increase revenue. The goal for the development is for the app to live on both iPhone and Android app stores.*

## Requirements

* *Multi-platform phone app*
* *Increase customer base*
* *Increase revenue*

## Design Constraints

* *Use a cross-app language such as react native that would enable Chat Away to develop once but deploy everywhere*
* *Incredibly smooth, intuitive, and well-researched UI.*
* *Easy to sign up and purchase subscriptions (assuming that is one of their main income models)*

## Rationale

* *Enabling Chat Away to be able to deploy a single app onto multiple device platforms would save them significantly when compared to specific apps/codebases for each specific platform. It also saves time when it comes to debugging as well as feature releases.*
* *It’s no secret that users require a smooth, intuitive, and well-researched user interface. When you think of the most popular apps, out of all the ideas, features, functionality they all have one thing in common. A clean, smart, intuitive, smooth user interface. There is likely to be some API functionality for the backend of the application, so we really need to focus a large amount of energy on the UI. Failure in executing the user interface, could make the application fail when released.*
* *Next to the UI it needs to be easy to sign up and make sense to purchase the subscription – assuming this is one of their main sources of revenue. This will be assisted heavily with the user interface.*

## Recommendations:

### Operating Platform:

*My recommendation would be to operate on a Windows server. One of the main reasons for this is it’s among the most widely used platforms for hosting applications. You can connect a seemingly endless number of applications and services to help monitor security, performance, application health and uptime metrics. Windows operating systems also are compatible with Java API which would enable the back-end code to be accessible via other platforms such as iOS, Android, etc.*

### Operating Systems Architectures:

*The Windows operating system has two kinds of system processes. One is the system process, and the other is the user process. System process has unrestricted access to manipulate hardware and operating system data. The user process can only access operating system data through sending requests through the system process. Numerous base processes such as system support, services, user applications, and environment subsystems get pushed to subsystem DLL’s. From there, system support services, services, and user applications are processed through Executive and then further moved through kernel and device drivers to end at HAL. Executive, kernel, device drivers, and HAL are all in kernel mode. The environment subsystems in user mode moves to the windowing and graphics part of kernel mode.*

### Storage Management:

*Included in the Windows operating system is disk management along with Storage Sense. Additionally, Windows has a disk cleanup tool. Disk management is a utility mostly used for advanced storage tasks. Disk cleanup and Storage Sense are useful in helping delete files that are unnecessary and that take up storage space.*

### Memory Management:

*Memory management is built into windows operating systems as a system utility. For the game images to be easily accessible by the application a logical folder structure with file locations stored in a database for quick and easy access.*

### Distributed Systems and Networks:

*A client-server distributing system would be the recommended choice as it would be easiest for the client to edit, add, and remove images onto the game and have them apply across the different instances of the application. Additionally, with different nodes (users) joining and leaving the network the client-server system would allow for a more stable connection and user experience with the application. The risk with the client-server system is when it comes to outages. Should the client-server system fail, unless there is redundancy and a readily available failover switch the application will suffer and go offline until the outage is resolved.*

### Security:

*In addition to tools such as Windows Defender, Windows Operating systems are compatible with countless security applications that range from encryption, NGINX, data processing and monitoring and more. There is no shortage of security configurations available on Windows Operating systems.*

## Evaluation

### Server Side

*The client asked to evaluate the three operating platforms in addition to mobile platforms. The three operating platforms are Linux, Mac, and Windows. All three offer the ability for them to run their application but I would recommend Linux or Windows due to compatibility with other software and hardware for monitoring, alerting, security, etc. ease of use, and overall supportability. Mac is generally not known for hosting so software and support could be limited should they decide to go this route. Linux and Windows also allow hosting of an API that would be accessible from all other mobile platforms should they build a user interface for it. Potential costs would be comparable should they deploy on Linux or windows. Though Linux is open source whereas windows operating systems do have fees to purchase newer versions.*

### Client Side

*Building software that is movable across devices is not impossible, in fact it’s quite a bit cheaper and more accessible than one would expect. Using front end frameworks such as React Native would enable the user to “build once and deploy everywhere” which is the entire mindset of using React Native as a user interface. The same code could be used for Android, iOS, Web, Windows phone, amazon fire tablet, and more. The only extra costs would be the costs of deploying an application to the respective phone app stores as well as having the physical device – or web tool that can emulate it – for validation testing. It would be a bit longer to develop this way for one single system, but it would be far shorter to do this than rebuilding the app for every unique system.*

### Development tools

*I’d recommend the following development tools: An IDE such as IntelliJ for Java API development as well as Visual Studio Code for React Native UI development. Both IDEs offer a free tier for the client to use without increasing costs. I would recommend a small team for API development as well as a team for UI development this would enable both front and backend to be developed at the same time and with dedicated focus, though it would be possible for a single individual or two to do both API and UI. I would recommend a physical device that connects to each app store they intend to release the app on for validation, such as an iPhone, iPad, Android phone and tablet, etc. if costs are tight, you can skip the tablet and just obtain a physical iPhone and Android phone. The web has capabilities that can accommodate the tablet screen resolutions.*