**D424 – Software Capstone**

**Task 2**



|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | Discord-Like Application |
| **Student Name:** | Jesse Tiemens |

**Table of Contents**

Contents

[**Table of Contents** 2](#_Toc152666055)

[**Business Problem** 2](#_Toc152666056)

[**The Customer** 2](#_Toc152666057)

[**Business Case** 3](#_Toc152666058)

[**Fulfillment** 3](#_Toc152666059)

[**SDLC Methodology** 3](#_Toc152666060)

[**Deliverables** 4](#_Toc152666061)

[**Project Deliverables** 4](#_Toc152666062)

[**Product Deliverables** 4](#_Toc152666063)

[**Deployment Plan and Outcomes** 4](#_Toc152666064)

[**Project Timeline** 5](#_Toc152666065)

[**Environments and Costs** 6](#_Toc152666066)

[**Programming Environment** 6](#_Toc152666067)

[**Environment Costs** 6](#_Toc152666068)

[**Human Resource Requirements** 6](#_Toc152666069)

[**Validation and Verification** 7](#_Toc152666070)

# **Business Problem**

**The Customer**

Our target customer is anyone who wants to create or be a part of online communities that facilitate communication. Our role will be to provide infrastructure to allow these customers to create online communities, or “servers”, on a global scale.

On the short term, we will be releasing a minimum viable product that will consist of the ability to create servers that have text channels, video channels, and audio channels, with the ability to invite users via a generated invite link, as well as the ability to give users roles that will help in moderation, ranging from guest to administrator, with a moderator role in between.

A long-term goal would be to provide more of the features that Discord, our reference point, already enables, such as adding users as friends, streaming your screen to servers/personal calls, and so on.

The growth we expect to experience is enormous if we can secure an initial client base. Discord has millions of users and is growing exponentially. We can hope to see the same exponential growth if we offer a competitive alternative platform for users to choose.

## **Business Case**

Finding or founding an online community that has robust communication features at the forefront can be difficult. Many existing platforms only offer a few communication features or are not fit to handle thousands of users concurrently.

For example, Facebook offers ways to connect between friends, and even join pages where users can create posts where users can communicate through a comment section. However, a major feature that Facebook lacks is a place where users can easily drop in and out of a public voice conference call.

Our application will offer this feature. Simply create or be invited to any server, which will function similarly to a public Facebook group/page. On the user interface will be a listing of the text, audio, and video channels that are available for the server. Interacting with the label for any channel will immediately join you into that conversation.

## **Fulfillment**

Our solution will initially be a web application hosted on a cloud application hosting service. Provided we experience significant user growth, we can expand to offering a standalone application for multiple platforms, including Android, iOS, Windows, Linux, and macOS. The web application will be hosted with a Next.JS 13 App Router web service. It will utilize Clerk as an authentication API, with a MySQL relational database hosted on PlanetScale. This database will store users, text messages, servers, and user roles.

Once a user registers, they will be able to authenticate with Clerk, and then be presented with a screen showing the servers they are a part of. In the server display on the left of the screen, there will always be an option to create a new server to easily facilitate creation of online communities. When a server is selected from the server display, next to the server display will be a list of channels that that server has. A channel can either be for text messages, audio conference calls, or video conference calls.

If a text channel is selected, the main section of the screen will show a conversation history with a text input at the bottom to send new messages. A message will be stored on the MySQL database with the time of the message’s sending being stored alongside the user who sent it, and the content of the message. The page will utilize an active web socket to be updated any time another user sends or edits a message. Provided this web socket fails, there will be an automatic polling fallback that will automatically check for new or updated messages every 1 second.

Video and audio channels will create or join a new audio/video conference call that is facilitated by the LiveKit API.

Server administrators will be able to create, delete, and rename any type of channel in a server, as well as invite users, or kick them from the server. They will also be able to assign roles to users already in the server. Users will join as a guest at default, who has no administrative permissions. Administrators can assign the moderator or administrator role to any user they choose. Moderators are allowed to kick users other than administrators or create any channel.

# **SDLC Methodology**

This project will be utilizing the Waterfall SDLC Methodology to develop a prototype, after the prototype is deployed, we will utilize a DevOps SDLC Methodology. It will be using waterfall for prototype development as the requirements are well understood and defined, given that we are using Discord, an existing application, as a starting point. After deployment, DevOps will serve updating a live service much better than any other methodology, as it will allow us to respond to user feedback with agility.

During the analysis phase, we will spend time studying the user interface of Discord, and brainstorm how the functionality of the application was facilitated on the backend.

During the requirements phase, we will identify the business case of our target customer base. We asked ourselves what the users liked about Discord, and why it has received such explosive growth. Our findings were listed in the previous sections of this document.

The design phase will have us creating wireframes of our user interface, and flowcharts of the user’s journey through using our application. We will also identify the tables and respective fields of said tables for facilitation of users, messages, and servers. The final aspect of our design phase will be identifying publicly available APIs that will ease our development and deployment processes. We identified PlanetScale as a suitable deployment solution for our MySQL relational database, and LiveKit for facilitating audio and video conference calls, something that would be very difficult to build from scratch.

While in the development phase, we will take the designs and start writing code to build the infrastructure necessary for the application to function.

Our testing and integration phase will consist of performing unit tests on features during development, performing actual full tests with a small group of friends once a feature-complete version is deployed.

Deployment of our application will be handled by Railway, a cloud application hosting service. This service provides GitHub integration and will automatically redeploy when any commit has been pushed to the master branch.

Maintenance will be an important phase for us. Since our initial testing after deployment will only consist of a small number of concurrent users, we will not be aware of issues that will inevitably crop up at hundreds and thousands of concurrent users. We will have technicians at standby at user count thresholds of 100 and 1,000, and plan for higher user counts once those thresholds have been met.

# **Deliverables**

## **Project Deliverables**

* Project Timeline
  + The order of development tasks, with dates and associated development cycle phases for each task.
* Testing Plans
  + Unit and regression testing plans for each aspect of the application.
* Object Relational Diagram
  + Diagram depicting how each object in the backend of the application links up to each other, and what properties each object possesses.

**Product Deliverables**

* Wireframes
  + We will develop low-fidelity visual representations of the various screens and modals for our proposed user interface.
* Prototype
  + The deliverable that is the focus of this project is a minimum viable product that will satisfy our target customer. This deliverable is the web application interface for our Discord-like application with the barebones features defined earlier in this document.

# **Deployment Plan and Outcomes**

The initial deployment of this application is simple. It will be handled by Railway, a cheap cloud application hosting solution. This approach will allow us to focus more on development, as the infrastructure and platform upkeep and procurement will be managed by a third party. It also will allow us to be flexible if our user count skyrockets. We will be able to employ elastic load balancers to divide server traffic between multiple instances running our application. Railway also has built-in CI/CD pipeline support so we can quickly push updates without impacting the application’s uptime.

After the prototype application has been deployed, as we switch to a DevOps development lifecycle, our team will be split into Development and Operations teams. The Operations team will be focused on deployment, configuration, and monitoring of the application while it is live. The Development team will be focused on writing code for new features in response to user feedback, as well as running verification tests before pushing code to the Operations team to deploy.

# 

# **Project Timeline**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Phase | Milestone/Task | Deliverable | Description | Dependencies | Resources | Dates |
| 1 | Analysis | Task 1 | Requirements | Analyzing Discord and identifying the core features necessary to create a minimum viable product. | None | 1 day of design team wages | 12/10/2023-  12/11/2023 |
| 2 | Design | Task 2 / Design files | Low Fidelity Wireframe | Analyze Discord through the lens of recreating its user interface, analyze how its components are laid out, and create a low fidelity representation of the user interface. | None | 1 day of design team wages | 12/11/2023-  12/12/2023 |
| 3 | Design | Task 2 / Design files | Object Relational Diagram | Diagram depicting how each object in the backend of the application links up to each other, and what properties each object possesses. | 2 | 1 day of development team wages | 12/12/2023-  12/13/2023 |
| 4 | Design | Task 3 / Design files | Testing Plans | Documents depicting unit and regression test plans, detailing expected results. | 3 | 1 day of development team wages | 12/13/2023-  12/14/2023 |
| 5 | Development | Task 3 | Prototype | Development of all code required to build a minimum viable product as defined in design documents. | 4 | 8 days of development team wages | 12/15/2023-  12/23/2023 |
| 6 | Deployment | Task 4 | Access to Prototype | Deploying the prototype to Railway so that users can have access to the application. | 5 | 1 day of development team wages | 12/23/2023  12/24/2023 |
| 7 | Maintenance | N/A | Continued application support | Splitting the project team into Development and Operations teams as defined in “Deployment Plans and Outcomes” section of this document. | 6 | Development and operations team wages for each day the application runs | 12/25/2023-  ? |

# **Environments and Costs**

## **Development Environment**

* Windows PC running the following software for each member of the development team: (~$500 for hardware and operating system for each employee, all software free.)
  + Visual Studio Code
  + Node.JS
  + Next.JS 13
  + Prisma MySQL database for testing environment

**Environment Costs**

The deployment option we have chosen has allowed us to leverage the cost benefits of a cloud application hosting solution. Our hosting environment consists of PlanetScale for our MySQL database host, which has a free hobbyist tier of hosting plan, with options to scale up infrastructure as necessary as the application’s customer base naturally grows. The other cloud solution we will be utilizing is Railway, which has a $5 per month hobby plan, with the ability to scale up just as PlanetScale offers.

Not only are these monthly fees incredibly low, but it also allows us to trade capital expense for variable expense, as we do not need to secure a physical location and purchase several servers to build our infrastructure from the ground up, and risk under-, or over-utilizing our servers. It also disallows us to be flexible to changing usage data.

## **Human Resource Requirements**

The wage resources for each step have previously been defined in the Project Timeline table earlier in this document. The following is a table depicting the project-wide sum of team member wages.

|  |  |
| --- | --- |
| Team | Total Wages |
| Design Team | 2 business days per team member |
| Development Team | 11 business days per team member during prototype development, then after deployment:  1 business day per team member per day that a development project is being worked on |
| Operations Team | 1 business day per team member per day of application being online |

# **Validation and Verification**

Our testing method will consist of three main testing components. Firstly, there will be an automated unit test for each bit of functionality that the application offers. Each automatic unit test will have an expected result defined in a table, and the test will be analyzed as a pass/failure, depending on whether the expected result is provided when the script runs. This will ensure that code is being developed that accomplishes the intended effect, according to the requirement documentation.

Secondly, regression testing will be completed upon completion of each module of the application to ensure that the completion of the new module does not impact the correct execution of previous modules. The regression testing will simply run the already defined unit tests for the other modules after the unit tests for the current module finishing development have finished executing. As these regression tests are simply running other unit tests, they will be judged in the same manner.

Finally, once a release candidate for the final prototype of the project is completed, we will get a small focus group of 3-4 friends who will provide their feedback as to whether they believe the application’s user experience to be sufficiently easy to use. If the feedback from our focus group is overwhelmingly negative, we may have to alter major aspects of the project before re-testing. This is an unlikely situation, as we will be using a pre-existing application that many users already find the user experience of satisfactory to begin with.