**OpinionMarket**

Josh Van de Walle

CST-451 Capstone Project Requirements Document

Grand Canyon University

Instructor: Professor Mark Reha

Revision: 1.2.0

Date: April 1, 2021

**ABSTRACT**

The moniker OpinionMarket captures the concept of a marketplace of ideas making it the perfect name for an application all about sharing information and thoughts. The application is a web platform for social news and community-driven discussion that will facilitate discourse between individuals based on common interests. OpinionMarket will allow users to join communities of their choice, post content in those communities, comment on content posted by others, upvote or downvote any post or comment, and send direct messages. These features will support the creation of a vast ecosystem of meaningful discussions.

The platform is designed to be easy for anyone to use and helpful for everyone. Whatever a user is interested in, be it a hobby, professional skill, or theoretical physics, they can find or start an online community dedicated to it. Community rules establish user-enforced behavioral regulations that keep communities focused on their topic. Customization features make communities unique, give them character, and allow them to stand out. The wealth of features they provide make OpinionMarket communities an excellent place to look for help, show off accomplishments, or find discussion of just about anything. The application presents an intuitive and friendly user interface that makes it easy to dive-in and start browsing content-rich communities.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 11/01/2020 | Josh Van de Walle | Initial version for review/discussion |
| 12/01/20 | Josh Van de Walle | Updated project name to OpinionMarket |
| 12/17/20 | Josh Van de Walle | Updated Logical System Design to indicate that Service Discovery is dynamic |
| 01/12/21 | Josh Van de Walle | Removed lingering reference to “Sententia” |
| 03/10/21 | Josh Van de Walle | Updated Logical System Design to indicate that compensation transactions are being used |
| 03/19/21 | Josh Van de Walle | Removed AWS Cloudwatch and Loggly from technical requirements |
| 04/01/21 | Josh Van de Walle | Centered all images |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Functional Requirements 5

Non-Functional Requirements 5

Technical Requirements 7

Logical System Design 8

User Interface Design 10

Appendix A References 26

Functional Requirements

**User Stories**

OpinionMarket’s functional requirements define what the application will do, but not how it will do it. Since these requirements relate to system functionality, all technical implementation details have been left out. Technical details will be addressed in the design phase of the OpinionMarket project. All functional requirements have been enumerated in the attached requirements.xslx spreadsheet, under the Discussion Community, User Accounts/Profiles, Authentication, Conversation, and Moderation tabs. Note that a full list of error messages referred to in the requirements can also be found in the requirements.xslx spreadsheet.

The table below lists functional requirement user stories that have been taken out of scope along with the justification for doing so. No requirements have been taken out of scope.

|  |  |  |
| --- | --- | --- |
| **Use Case or User Story** | **Approval Date** | **Justification** |
|  |  |  |

Non-Functional Requirements

**User Stories**

OpinionMarket’s non-functional requirements (NFRs) form constraints around the application’s behavior and technical design. All code must be designed and written to meet the NFRs, and application testing should include testing them. The Sententia project has two NFRs as shown in user story form in the table below. Note that a full list of error messages referred to in the requirements can be found in the requirements.xslx spreadsheet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue Key** | **As a/an <actor>** | **I would like to <action>** | **So that <outcome>** |
| CAP-239 | As a system, | As a system, I want to prevent a user who is not logged in from accessing any secured resource, | so that I can ensure the confidentiality of sensitive information I store |
| CAP-238 | As a system, | I want to display error message #46 if any required database is offline when an attempt is made to access it regardless of the request, | so that I can hide my technologies |

OpinionMarket’s NFRs are also enumerated under the ‘Global’ tab found in the attached requirements.csv spreadsheet.

The following table lists NFR user stories that have been taken out of scope along with the justification for doing so. No NFR user stories have been taken out of scope.

|  |  |  |
| --- | --- | --- |
| **User Story** | **Approval Date** | **Justification** |
|  |  |  |

Technical Requirements

OpinionMarket will be built using state-of-the-art technology including frameworks, libraries, languages, database technology, containerization technology, cloud solutions, and development environments that are relevant in the software engineering industry. The following list outlines all the technologies that will be used including the relevant version. Version numbers are not included for cloud technologies because their versions are outside of OpinionMarket’s control.

* Spring Boot 2.3.5
* Java 1.8.0
* React 16.13.1
* Redux 4.0.5
* Material-UI 4.11.0
* Axios 0.20.0
* React-Router 5.2.0
* React-Router-DOM 5.2.0
* React-Images-Upload 1.2.8
* React-File-Uploader 1.0.0
* JavaScript ECMAScript 2018
* MongoDB 4.2.10
* MongoDB Atlas
* Docker Engine 19.03.13
* Spring Tool Suite 4.7.2.RELEASE
* Visual Studio Code 1.50.1
* AWS EC2
* AWS Elastic Beanstalk
* Swagger OpenAPI Specification 3.0.3
* Postman 7.34.0
* MongoDB Compass Community 1.21.2

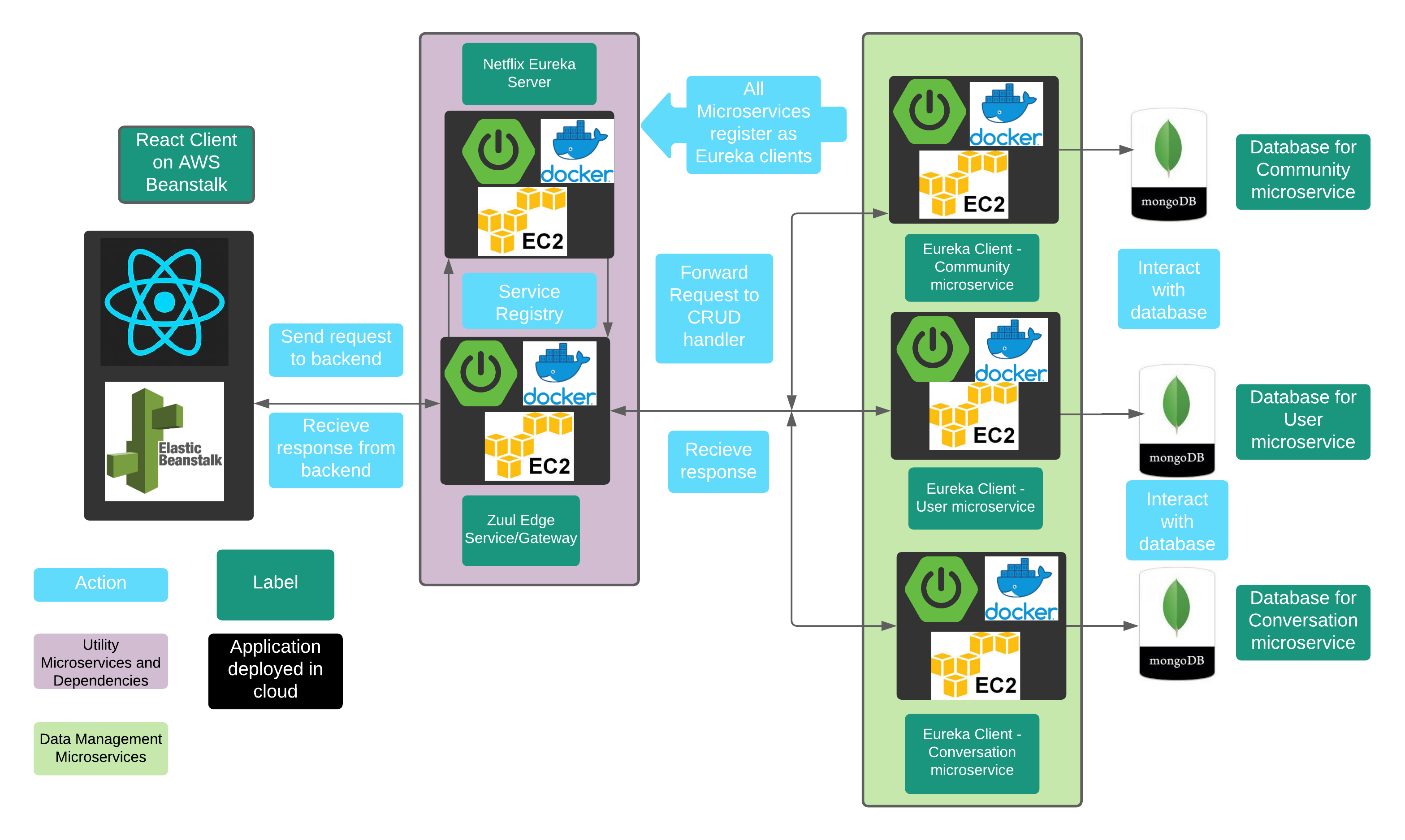
Logical System Design

**Introduction**

OpinionMarket will be designed using a microservice architecture. This architecture solves business problems but creates technical problems (Cloud Native Computing Foundation, 2017). Due to the complex nature of the expected OpinionMarket codebase, it is critical to have a solid understanding of the high-level systems and components that will comprise the final product.

**Solution**

Microservice architecture divides an application into smaller applications to increase the maintainability, technical flexibility, scalability, and fault-tolerance of the overall system (Nemer, 2019). The block diagram below illustrates the expected components, including Spring Boot microservices, Mongo databases, cloud solutions, containerization technology, third-party dependencies, and clients that constitute the OpinionMarket system.



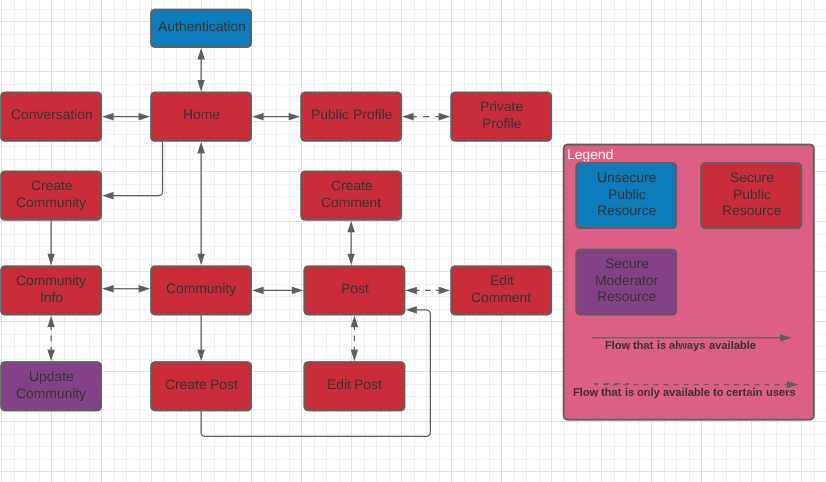
Two microservices to note are the discovery service (Netflix Eureka Server) and Edge Service/Gateway (Zuul). These microservices are utility services that oversee interaction with the data management microservices. The gateway recieves all client API calls and leverages the discovery service to call the appropriate data management microservice(s). Service discovery will be dynamic. Service orchestration, including compensation transaction management, is handled in the API gateway. Every microservice registers with the eureka discovery service as a client (Hossain, 2020). This allows the discovery server to know what data management microservice needs to be called for a given API request. The data management microservices are Spring Boot applications that perform create, read, update, and delete (CRUD) actions on their database. Note that each data management microservice is containerized using Docker, and has its own database. This means that full data consisitency is eventual. Compensation transactions are leveraged to kepp data consistent. The databases are hosted in the cloud on MongoDB Atlas. AWS has been selected as the cloud solution for OpinionMarket. AWS Elastic Compute (EC2) services will be used for the Spring Boot microservices, while Elastic Beanstalk will be used to host the frontend React application.

The microservice architecture has several advantages. Changes made to one microservice do not force the entire application to be re-compiled, meaning the code is easier to maintain. Maintainability also benefits from microservices being more simple, from a code standpoint, than monolith applications. This makes a microservice easier for a developer to understand when designing and building an update. Microservices can use different frameworks and databases based on what technology provides the best solution for the problem they are solving. This technical flexibilty means development teams are no longer forced to use technologies unsuitable to the task they are working on. Microservices make scaling up an application easier. Since services are separate it’s easier to scale only the services that must be scaled, improving efficiency. If there is a problem in one microservice the rest of the system is far less likely to be affected than with a monolith architecture because of the code boundaries between microservices.

Final decisions have now been made to use Spring Boot, MongoDB, React, Redux, Material-UI, Docker, and AWS in the OpinionMarket project. A final decision has also been made to use a microservice architecture in the project. If any technology or architecture pattern must be abandoned, the appropriate contingency plan from the project proposal will be implemented.

**User Interface Design**

The following image shows the OpinionMarket sitemap. The legend details the security level and availability of flows. Note that flows that are not always available (dashed lines) represent cases where only some authenticated users have permission to perform an action such as editing a community, editing a post, editing a comment, or editing a profile.



For the requirements phase of the project, rough wireframe concepts have been developed for the planned OpinionMarket UI. These initial concepts aided in writing the functional requirements for the system. The concept wireframes, however, should not be interpreted as final design plans. The final design for OpinionMarket will be delivered at the conclusion of the design phase of the SDLC. NOTE the former name of the OpinionMarket project was Sententia.

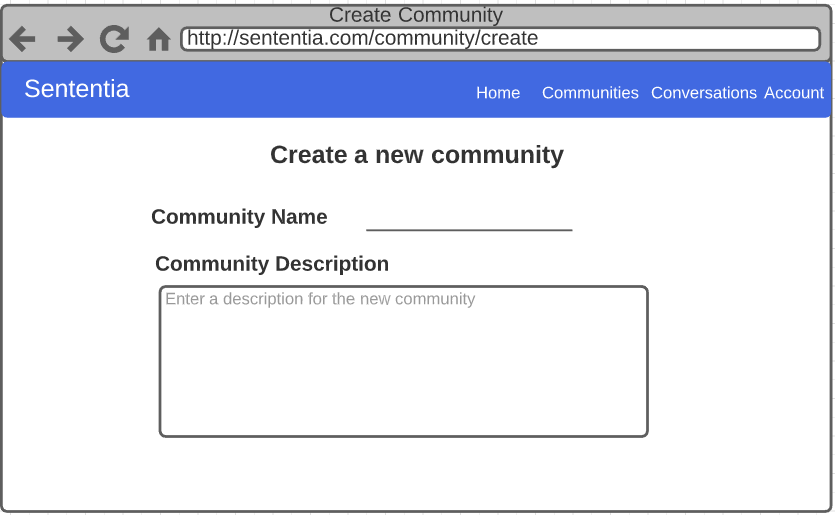
The rough concept for OpinionMarket’s authentication interface is shown below. On the authentication screen, Users can register or login.



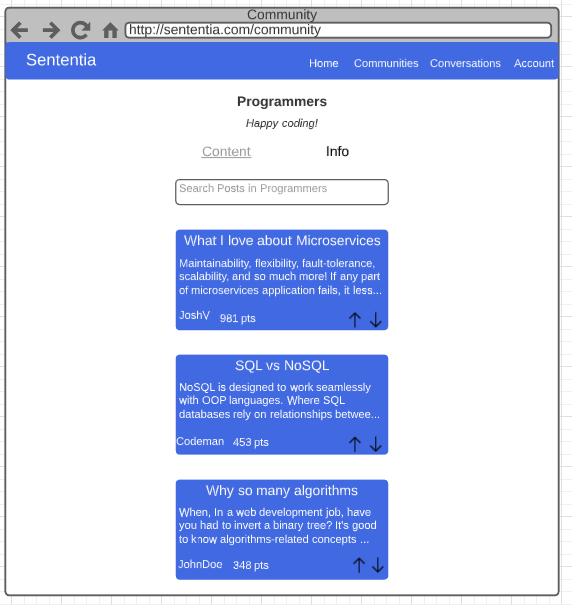
The rough concept for Sententia’s Home screen. On the home, screen, users, can search for communities, view communities, and create a new community.



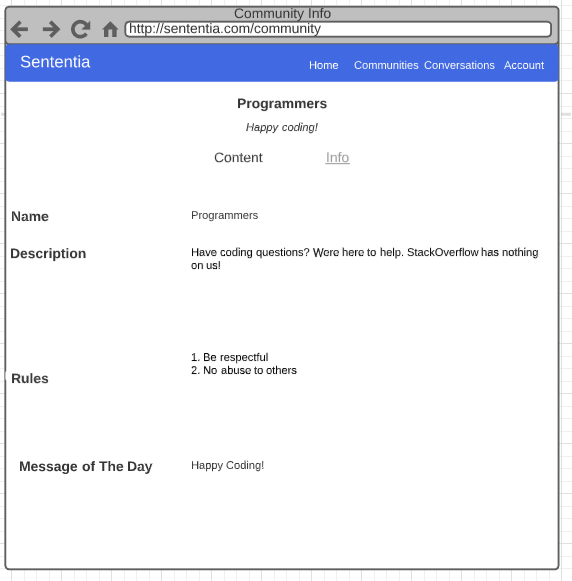
The rough concept for Sententia’s create community screen is shown below. On this screen, users can create a community by providing a community name and description.



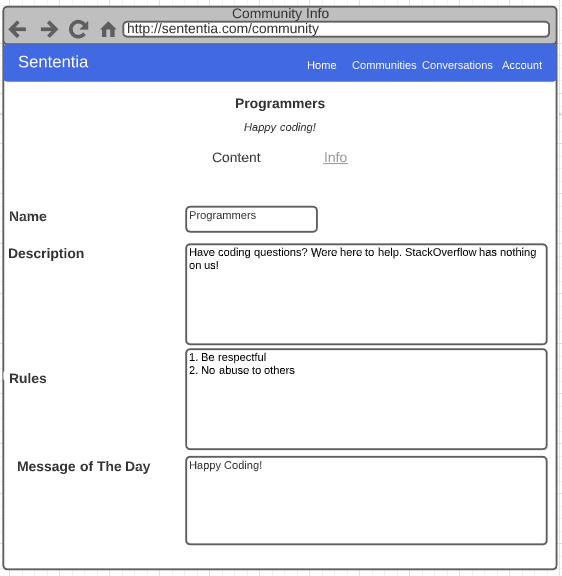
The rough concept for the community screen is shown below. On this screen, users can search for posts, view posts, see the community’s message of the day, upvote a post, downvote a post, and navigate to the community’s info page.



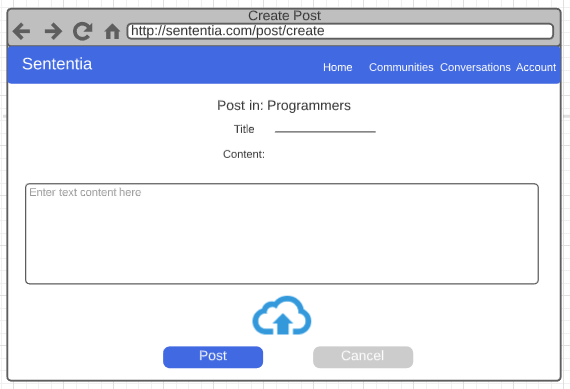
The rough concept for Sententia’s community info page is shown below. On this screen, users can see name, description, rules, and message of the day. They cannot make any edits unless they are moderators.



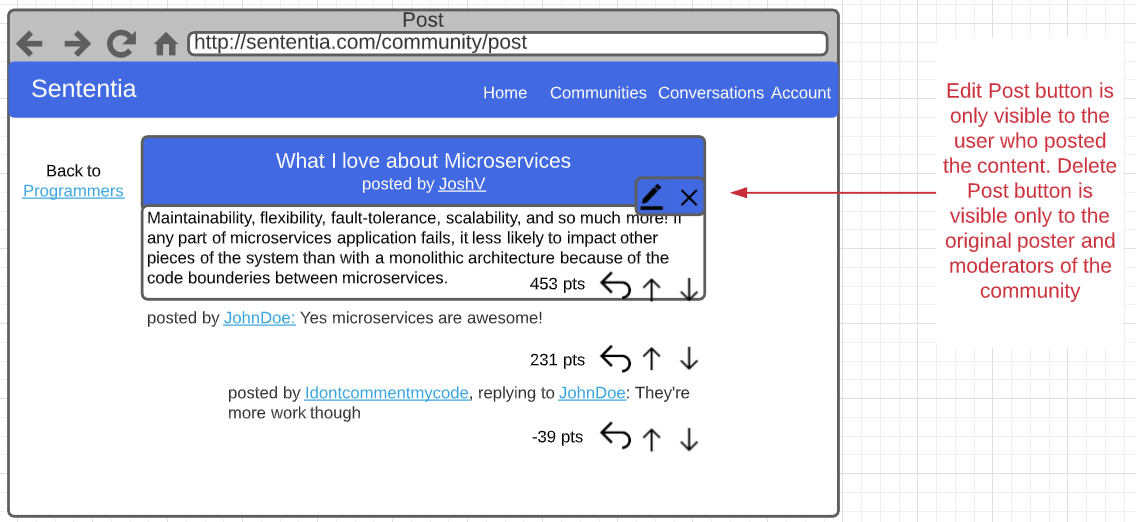
The rough concept for the edit community screen is shown below. Only moderators can see this screen. On this screen, moderators can edit the community name, description, rules, and message of the day.



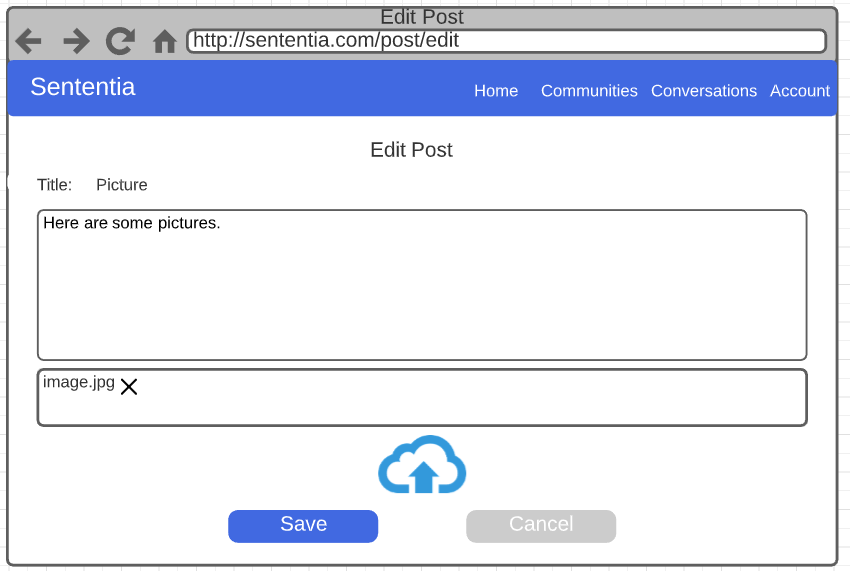
The rough concept for the Create Post screen is shown below. On this screen users can create a post including text, image, or video content.



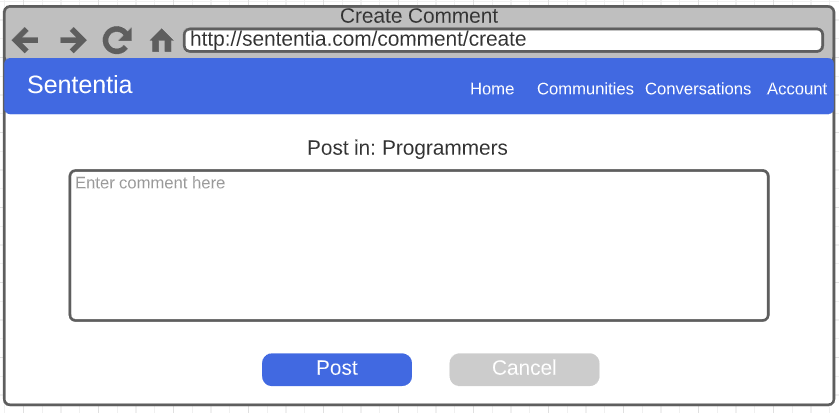
The rough concept for the Post screen is shown below. On this screen, users can view a post, view all comments on the post, upvote the post, downvote the post, upvote a comment, downvote comment, leave a comment on the post, and leave a comment on another comment. If the user is the original poster of the content, they can edit or delete the post. If the user is a moderator of the community, they can delete, but not edit, the post.



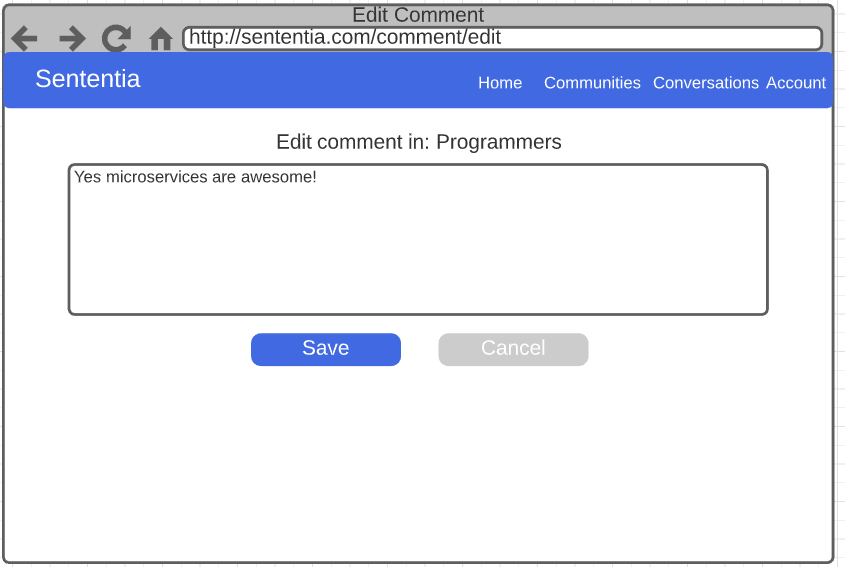
The rough concept for the Edit Post screen is shown below. On this screen, users can edit the text, image, and video content they have posted.



The rough concept for the Create Comment screen is shown below. On this screen, users can either create a comment or cancel the action.



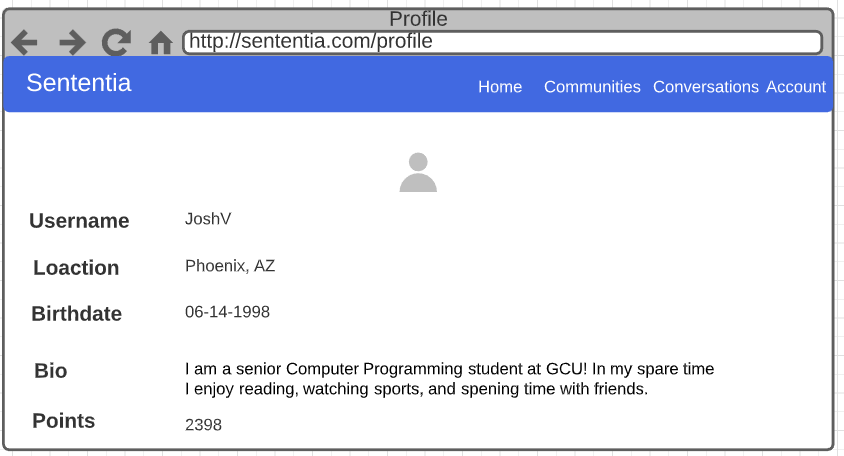
The rough concept for the Edit Comment screen is shown below. On this screen user can either edit their comment or cancel the action.



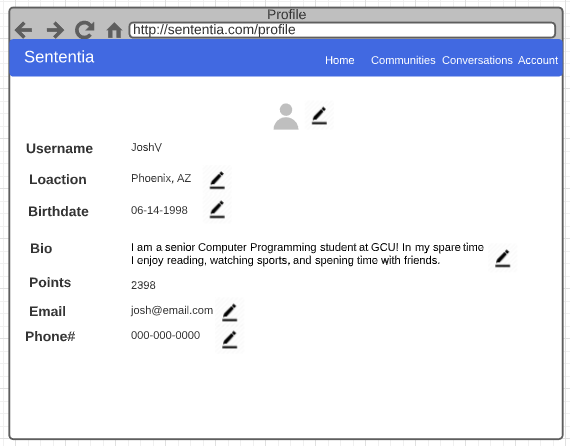
The rough concept of the Conversation screen is shown below. On this screen users can start conversation, select an existing conversation to continue it, send messages, edit messages they sent, and delete their messages they sent. Message sending and editing will be handled in the conversation screen. Consequently, no separate screens are planned for message creation or editing.



The rough concept for the Sententia public profile is shown below. On this screen users may view the public profile of another user.



The rough concept of the private user profile screen is shown below. On this screen, users can view and edit their own profile information and image.



**Change Controls**

The change control log records changes to this document.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Change Control Log | | | | | | | | | |
| **ID** | **Change Description** | **Priority** | **Originator** | **Date Entered** | **Date Assigned** | **Evaluator** | **Status** | **Date of Decision** | **Included in Rev. #** |
| 1 | Initial version for review/discussion | High | Josh Van de Walle | 11/01/20 | 09/28/20 | Josh Van de Walle | Done | 11/01/20 | 1.0.0 |
| 2 | Updated project name to OpinionMarket | High | Josh Van de Walle | 12/01/20 | 11/29/20 | Josh Van de Walle | Done | 12/01/20 | 1.1.0 |
| 3 | Updated Logical System Design to indicate that Service Discovery is dynamic | Medium | Josh Van de Walle | 12/17/20 | 12/17/20 | Josh Van de Walle | Done | 12/17/20 | 1.1.1 |
| 4 | Removed lingering reference to “Sententia” | Low | Josh Van de Walle | 01/12/21 | 01/12/21 | Josh Van de Walle | Done | 01/12/21 | 1.1.2 |
| 5 | Updated Logical System Design to indicate that compensation transactions are being used | Medium | Josh Van de Walle | 03/10/21 | 3/10/21 | Josh Van de Walle | Done | 03/10/21 | 1.1.3 |
| 6 | Removed AWS Cloudwatch and Loggly from technical requirements | Medium | Josh Van de Walle | 03/19/21 | 03/19/21 | Josh Van de Walle | Done | 03/19/21 | 1.1.4 |
| 7 | Centered all images | High | Josh Van de Walle | 04/01/21 | 04/01/21 | Josh Van de Walle | Done | 04/01/21 | 1.2.0 |

Appendix A – References

# Cloud Native Computing Foundation. (2017, April 10). *Go + Microservices = Go Kit [I] – Peter*

# *Bourgon, Go Kit.* Retrieved from

# https://www.youtube.com/watch?v=NX0sHF8ZZgw&feature=youtu.be

Hossain, A. (2020, September 17). How to Implement Oauth2 Security in Microservices DZone.

https://dzone.com/articles/how-to-achieve-oauth2-security-in-microservices-di

Nemer, J. (2019, November 13). *What are microservices?* Cloud Academy.

https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/