OpinionMarket

Josh Van de Walle

CST-451 Capstone Project Proposal

Grand Canyon University

Instructor: Professor Mark Reha

Revision: 1.1.9

Date: 3/31/2021

**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** |
| 9/27/20 | Josh Van de Walle | Initial version |
| 9/29/20 | Josh Van de Walle | Changed ‘app’ to ‘application’ in all sections |
| 12/1/20 | Josh Van de Walle | Updated project name to OpinionMarket |
| 12/15/20 | Josh Van de Walle | Added service discovery to assumptions and constraints list and issues log |
| 02/08/21 | Josh Van de Walle | Added service orchestration to constraints list and issues log |
| 02/14/21 | Josh Van de Walle | Added log aggregation to issues log and list of risks |
| 02/27/21 | Josh Van de Walle | Added client application responsiveness as a constraint |
| 03/09/21 | Josh Van de Walle | Updated High-level solution to include compensation transactions |
| 3/10/21 | Josh Van de Walle | Removed lingering reference to “Sententia” |
| 3/12/21 | Josh Van de Walle | Added “More community customization options” to out of scope features list |
| 3/15/21 | Josh Van de Walle | Updated Work Breakdown Structure to reflect work done |
| 03/31/21 | Josh Van de Walle | Updated Assumptions & Constraints list to reflect that resolved assumptions and constraints |

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

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

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**Project Approval**

Professor Mark Reha

**TABLE OF CONTENTS**

Project Overview and Project Objectives 5

Project Scope 7

Project Success Measures 10

Project High-Level Solution 12

Project Controls 14

Appendix A – References 18

Appendix B – Copyright Compliance 19

Project Overview and Project Objectives

**State the Problem and Background**

People need a place to participate in community-driven discussion about the things that matter most to them. OpinionMarket aims to be that place. All public discussion on OpinionMarket will take place in a community specific to the topic. In this way, OpinionMarket will bring individuals together from all around the world based on their shared interests. The existence of this market space has already been proven by the commercially successful application, Reddit. Existing platforms, however, leave much to be desired in the realms of User Interface (UI) design and community environment. Reddit specifically has a cluttered UI and lacks a strong sense of community. OpinionMarket, in contrast, will emphasize community and be easy to use.

OpinionMarket will achieve the goal of making an easy-to-use application by committing to a UI design that is clean, intuitive, responsive, and to-the-point. Cluttered user interfaces overwhelm users with excessive and little-used options that steepen the learning curve for the application. Additionally, the OpinionMarket UI will be self-explanatory removing as many obstacles as possible from the user onboarding and retention processes. This will maximize the potential user base and thereby set OpinionMarket up for success in both the short and long term.

Equally important to OpinionMarket’s success will be the community-oriented design of its User Experience (UX). Competitors, such as Reddit, ironically fail to create a strong sense of community in their implementation of the community concept. OpinionMarket will differentiate itself from the competition by placing importance on recognition and familiarity within communities. Users whose content is well-received will be acknowledged and message of the day posts will create a feeling of familiarity. Additionally, communities will include customization and moderation features both of which reinforce the sense of familiarity OpinionMarket strives to achieve.

OpinionMarket’s intuitive UI and community oriented UX provide the project with a competitive advantage in a proven market. The application will be easily accessible over the web by any HTTPS-capable device. Taken together, the simple UI and ease of access provided by Sententia will propel the project to success.

**Project Objectives**

If the OpinionMarket project is a success, I will have developed a robust social news application. The application should implement standard social news and community-driven discussion features with an eye to creating an intuitive UI and community-oriented UX. High level features will include:

* Discussion Communities
* The ability to post content in a community
* The ability to reply to comments on posts or comments
* Voting (ability to like or dislike content)
* User profiles
* Direct message conversations
* User recognition features (such as a top user list or leaderboards)
* A clean and intuitive UI

A comprehensive list of in-scope features can be found under the Project Scope section.

**Challenges**

Like any software development project, OpinionMarket presents challenges that must be identified at the outset. The technology and code design I will use to build the application is state-of-the-art but also unfamiliar presenting significant technical hurdles. My limited business experience makes the project prone to organizational obstacles. High-level challenges include:

* Use of unfamiliar technologies
* Use of unfamiliar design patterns
* Inexperience with early phases of the SDLC

A more detailed list of project risks can be found in the Project Controls section of this document. Identifying these challenges early in the project provides me with the maximum amount of time to mitigate them and ensure they do not prevent the successful achievement of the project objectives.

**Benefits and Opportunities**

OpinionMarket will be a robust application built with frameworks, libraries, tools, and design patterns that are frequently used in the software development industry. Building the application will give me useful expertise that I can leverage in the workplace, on my resume, and in my LinkedIn profile. The project also provides an expandable platform that I can build on to learn even more industry-relevant technologies. I could learn the popular framework React Native by adding a mobile application to the project, for instance.

In addition to the technological benefits of OpinionMarket, the project will also increase my understanding of the Systems Development Life Cycle (SDLC). At the outset of the project, I was largely unfamiliar with the initiation, system concept and development, planning, and requirements analysis phases of the SDLC. After managing those phases of the OpinionMarket project, I will be much more well-rounded as a professional software developer, increasing opportunities for advancement beyond entry-level work. That experience will also be useful if I decide to start my own business. Building OpinionMarket will broaden my professional horizons and increase my chances of obtaining a full-time position as a software engineer.

Project Scope

The scope of the project is to build a web application consisting of a client app and server-side app that support user accounts, communities, posts, comments, and conversations.

The following are in-scope features:

* A set of enterprise backend APIs
* A client application
* User accounts and authentication
* Public user profiles
* Discussion communities
* Community moderation
* Community customization
* Posting text in communities
* Posting images in communities
* Commenting on posts
* Commenting on comments
* Private conversations between users
* Upvoting posts
* Downvoting posts
* Upvoting comments
* Downvoting comments

The following are out-of-scope features:

* A mobile application
* Option to switch between light mode and dark mode
* Group conversations
* Leaderboards that rank users, communities, posts, and comments, by points
* Reddit-style awards
* Bots that automate tasks such as moderation
* User saving a post
* What You See Is What You Get (WYSIWYG) for text in posts
* Posting videos in communities
* More community customization options (color schemes, customized voting icons, etc.)

The table below is a high-level schedule for the OpinionMarket project, including planning, requirements analysis, technical design, development, and testing tasks.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Work Breakdown Structure | | | | | | | | | | |
| ID | Task | Dependencies | Status | Effort Hours | Cost | Start Date | Planned Completion | Estimate to Completion | Actual Completion | Resource |
| 1 | Complete proof of concept application built with MongoDB and Express | Completion of tutorials (See resource column) | Done | 4 | N/A | 09/07/20 | 09/07/20 | 09/07/20 | 09/07/20 | https://zellwk.com/blog/crud-express-mongodb/ |
| 2 | Complete project proposal document and lean canvas | None | Done | 15 | N/A | 09/09/20 | 09/27/20 | 09/27/20 | 09/27/20 | N/A |
| 3 | Complete proof of concept application built with a microservice architecture, MongoDB, Spring Boot, React, Redux, and Material-UI | Completion of tutorials (see resource column) | Done | 40 | N/A | 09/20/20 | 10/10/20 | 10/20/20 | 10/29/20 | CST-339 Activities 5-8 and  https://www.springboottutorial.com/creating-microservices-with-spring-boot-part-1-getting-started |
| 4 | Fully define project requirements in Project Requirements Document | Completion of project proposal | Done | 20 | N/A | 09/28/20 | 11/01/20 | 11/01/20 | 11/01/20 | https://www.boost.co.nz/blog/2012/01/use-cases-or-user-stories  https://www.mountaingoatsoftware.com/agile/user-stories  “Write Good User Stories” from course padlet |
| 5 | Complete final architectural plan | Completion of Project Requirements Document | Done | 60 | N/A | 11/02/20 | 11/29/20 | 11/29/20 | 11/29/20 | https://skillsmatter.com/skillscasts/1325-simon-brown-architecture-where-do-you-start  “Web Application Design 101” from Course padlet |
| 6 | Complete Development of backend APIs, and microservices | Completion of the final architectural plan | Done | 60 | N/A | 11/30/20 | 02/21/21 | 12/20/20 | N/A | N/A |
| 7 | Complete client application | Completion of the final architectural plan | Not started | 60 | N/A | 12/21/20 | 03/06/21 | 1/30/21 | N/A | N/A |
| 8 | Complete testing phase | Completion of coding | Not started | 100 | N/A | 1/31/20 | 04/04/21 | 03/06/20 | N/A | N/A |
| 9 | Release product | Completion of testing phase | Not started | 20 | N/A | 03/07/20 | 04/04/21 | 04/04/21 | N/A | N/A |

Project Success Measures

The criteria by which the project’s success will be judged are enumerated in the table below.

|  |
| --- |
| Project Completion Criteria |
| 1 – 95% of requirements will be met |
| 2 – I will learn Spring Boot |
| 3 – I will learn to develop applications using a microservice architecture |
| 4 – I will learn React |
| 5 – I will learn MongoDB |
| 6 – I will learn Docker |
| 7 – I will learn Redux |
| 8 – I will learn Material-UI |

The table below lists project assumptions and constraints. Assumptions are anything that is expected to be true but may not be. Constraints are any limiting factors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Assumptions and Constraints | | | | | |
| ID | Description | Comments | Type | Status | Date Entered |
| 1 | Project will use Spring Boot | Proof of concept complete | Assumption | Resolved | 9/24/20 |
| 2 | Project will use React | Proof of concept complete | Assumption | Resolved | 9/24/20 |
| 3 | Project will use Redux | Proof of concept complete | Assumption | Resolved | 9/24/20 |
| 4 | Project will use Material-UI | Proof of concept complete | Assumption | Resolved | 9/24/20 |
| 5 | Project will use Docker | Proof of concept coming soon | Assumption | Resolved | 9/24/20 |
| 6 | Project will use a free cloud solution | AWS, Azure, and Heroku are under consideration | Assumption | Resolved | 9/24/20 |
| 7 | App must be configured to use third-party OAuth provider | Proof of concept with Github complete | Constraint | Resolved | 9/24/20 |
| 8 | Project requirements must be fully specified by 11/1/20 | None | Constraint | Resolved | 9/24/20 |
| 9 | Full technical design specifications must be complete by 11/29/20 | None | Constraint | Resolved | 9/24/20 |
| 10 | Backend Spring Boot microservices must be delivered by 02/21/21 | None | Constraint | Resolved | 09/24/20 |
| 11 | Unit and Integration testing must be accomplished by 04/04/21 | None | Constraint | Resolved | 09/24/20 |
| 12 | OpinionMarket must be completed and released by 04/04/21 | None | Constraint | Resolved | 9/24/20 |
| 13 | Microservices should call each other as rarely as possible | If microservices call each other too often they should be merged | Constraint | Resolved | 9/27/20 |
| 14 | Service Discovery required for microservices architecture | Service discovery should be handled dynamically on the server side. | Constraint | Resolved | 12/15/20 |
| 15 | Service/API Orchestration needed for voting use cases | Data consistency must be maintained through compensation transactions | Constraint | Resolved | 02/08/21 |
| 16 | Client application must be responsive for mobile devices | None | Constraint | Resolved | 02/27/21 |

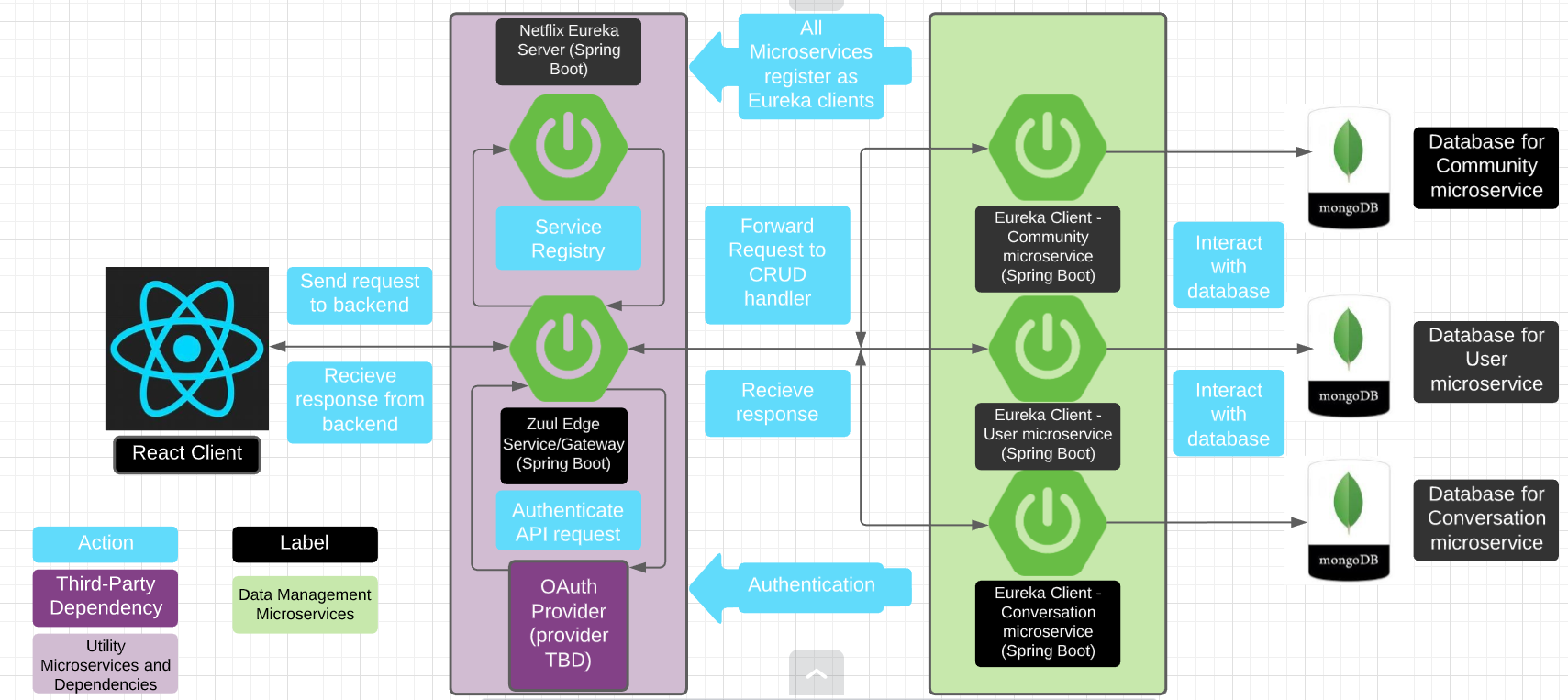
Project High-Level Solution

**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, 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. Every microserice 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 has its own database. This means that full data consisitency is eventual. Compenstation transactions will be used to maintain data consistency. Cloud solutions are not included in the diagram. Once a cloud provider is selected for the OpinionMarket project, the diagram will be updated to reflect the choice.

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 bounderies between microservices.

Final decsions on whether to use Spring Boot, MongoDB, React, Redux, Material-UI and a microservice architecture will be made by October 4. Final Decsions on whther to use Docker and AWS will be made by October 12. If any technology or architecture pattern is abandoned the appropriate contingency plan will be implemented. Contingency plans can be found in the Risk Management table below.

Project Controls

This section provides information about project risks, issues, and changes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Management | | | | |
|  | **Risk Probability** | **Risk Impact** |  |  |
| **Event Risk** | **(high, medium, low)** | **Risk Mitigation** | **Contingency Plan** |
| What is the risk? | What is the probability? | What is the impact if the risk occurs? | What can be done to minimize the risk? | What can be done to minimize the impact of the risk? |
| Use of unfamiliar technology, Spring Boot, delays the project | High | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept in Spring Boot will help to uncover issues before they effect the project. The following were used: https://www.dineshonjava.com/developing-your-first-spring-boot-application-hello-world/ and http://websystique.com/spring-boot/spring-boot-rest-api-example/#:~:text=Rest%20API%20with%20Spring%20Boot%20is%20no-different%20than,time%20by%20many-fold%2C%20certainly%20worth%20giving%20a%20try and https://www.springboottutorial.com/creating-microservices-with-spring-boot-part-1-getting-started | If Spring Boot proves unsuitable for use, .NET Core will be used instead. |
| Use of unfamiliar technology, MongoDB, delays the project | High | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept in MongoDB will help to uncover issues before they effect the project. Tutorials used: https://www.tutorialspoint.com/mongodb/index.htm | If MongoDB proves unsuitable for use, PostgreSQLwill be used instead. A final decision to proceed with MongoDB has already been made. |
| Use of unfamiliar technology, React, delays the project | High | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept in React will help to uncover issues before they effect the project. Tutorials used: https://www.youtube.com/watch?v=NemyDIUcC64 | If React proves unsuitable for use, the client-side app will be built with Angular instead. |
| Use of unfamiliar technology, Redux, delays the project | Medium | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept using Redux will help to uncover issues before they effect the project. Tutorials used:  https://www.youtube.com/watch?v=NemyDIUcC64 | If Redux with React proves unsuitable for use, the client-side app will be built with Angular instead. |
| Use of unfamiliar technology, Material-UI, delays the project | Medium | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept using Material-UI will help to uncover issues before they effect the project. Tutorials used:  https://www.youtube.com/watch?v=NemyDIUcC64 | If Material-UI proves unsuitable for use, Materialize will be used instead. |
| Use of unfamiliar technology, Docker, delays the project | Low | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept in Docker will help to uncover issues before they effect the project. Resources used: Extra Activity – Learning Docker and Docker for Mac/Windows (CST-323) | If Docker proves unsuitable for use, then the microsevices will not be stored in Docker containers. |
| Use of unfamiliar architecture, mircoservices, delays the project. | Low | The project is delayed to such an extent that success measures are not achieved, and in-scope features are not finished. | Completion of tutorials and proofs of concept for microservices will help to uncover issues before they effect the project. Tutorials used: https://www.springboottutorial.com/creating-microservices-with-spring-boot-part-1-getting-started | If microservices prove unsuitable, a monolith-style application will be used instead. |
| CORS must be enabled in Spring Boot to communicate with React | Low | Learning how to enable CORS in Spring Boot takes so long it delays the project. | A tutorial from the web will be used to obtain the knowledge of how to enable CORS in Spring Boot. Tutorial found: https://www.javadevjournal.com/spring-boot/spring-boot-cors/#:~:text=%20%20%201%20Controller%20Method%0ATo%20enable%20the,think...%203%20Spring%20CORS%20using%20Filter%20More%20 | In the unlikely event that CORS cannot be enabled in Spring Boot either another backend framework will be used or Spring Boot will be used to generate views in place of a front-end React app. CORS will be enabled in a Spring Boot proof of concept by 10/4/20 |
| COVID-19 | Medium | COVID-19 infection stops work on the project | Wearing masks and social distancing | Work will continue from quarantine environment |
| Cannot aggregate logs | Low | Visibility into application usage is limited making it harder to uncover bugs | Research log aggregation tools such as Loggly. | Debug using debuggers and tests |

The Issues Log tracks problems that are currently delaying the project. These are risks that were not mitigated in time. There no issues at present.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues Log | | | | | | | | |
| **ID** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| 1 | Services discovery unimplemented in the API gateway | If this issue goes unresolved, the client application will have to be service-aware | The Eureka tool from Netflix will be leveraged to implement service discovery dynamically on the server-side. | Josh Van de Walle | High | 12/15/20 | 12/20/20 | 12/20/20 |
| 2 | Fully guaranteed atomic transactions impossible | In certain scenarios data could become inconsistent | Compensation transactions will be implemented to mitigate this risk | Josh Van de Walle | High | 02/08/21 | 03/07/21 | 03/07/21 |
| 3 | Logs are not aggregated | Decreased visibility into application usage making bugs harder to find | Leverage a log aggregator such as Loggly | Josh Van de Walle | Medium | 02/14/21 | 04/04/21 | Unresolved |

Final decisions on project technologies and general architecture will be made before requirements and technical design specifications are written. Specifically, final decisions on whether Spring Boot, React, Redux, Material-UI, Docker, and a microservice architecture will be used are due by October 4, 2020. If any planned technology or architectural design are abandoned, then the appropriate contingency plan will be employed. Contingency plans can be found in the Risk Management Table.

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 | Replaced “app” with application in document | Low | Josh Van de Walle | 09/29/20 | 09/27/20 | Josh Van de Walle | Done | 09/29/20 | 1.0.1 |
| 2 | Changed name of project to OpinionMarket and updated Work Breakdown Schedule | High | Josh Van de Walle | 12/01/20 | 11/29/20 | Josh Van de Walle | Done | 12/01/20 | 1.1.0 |
| 3 | Added service discovery to assumptions and constraints list and issues log | Medium | Josh Van de Walle | 12/15/20 | 12/15/20 | Josh Van de Walle | Done | 12/15/20 | 1.1.1 |
| 4 | Added service orchestration to constraints list and issues log | Medium | Josh Van de Walle | 02/08/21 | 02/08/21 | Josh Van de Walle | Done | 02/08/21 | 1.1.2 |
| 5 | Added log aggregation to issues log and list of risks | Medium | Josh Van de Walle | 02/14/21 | 02/14/21 | Josh Van de Walle | Done | 02/14/21 | 1.1.3 |
| 6 | Added client application responsiveness as a constraint | Medium | Josh Van de Walle | 02/27/21 | 02/27/21 | Josh Van de Walle | Done | 02/27/21 | 1.1.4 |
| 7 | Updated High-level solution to include compensation transactions | High | Josh Van de Walle | 03/09/21 | 03/09/21 | Josh Van de Walle | Done | 03/09/21 | 1.1.5 |
| 8 | Removed lingering reference to “Sententia” | Low | Josh Van de Walle | 03/10/21 | 03/10/21 | Josh Van de Walle | Done | 03/10/21 | 1.1.6 |
| 9 | Added “More community customization options” to out of scope features list | Low | Josh Van de Walle | 03/12/21 | 03/12/21 | Josh Van de Walle | Done | 03/12/21 | 1.1.7 |
| 10 | Updated Work Breakdown Structure to reflect work done | Medium | Josh Van de Walle | 03/15/21 | 03/15/21 | Josh Van de Walle | Done | 03/15/21 | 1.1.8 |
| 11 | Updated Assumptions & Constraints list to reflect that resolved assumptions and constraints | Medium | Josh Van de Walle | 03/31/21 | 03/31/21 | Josh Van de Walle | Done | 03/31/21 | 1.1.9 |

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

Docker. (2020). *Components & Licenses.* Retrieved from

https://www.docker.com/legal/components-licenses

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

Facebook. (2018, September 7) *facebook/react.* Retrieved from

https://github.com/facebook/react/blob/master/LICENSE

Material-UI. (n.d.) *Material-UI* Retrieved from https://v3.material-ui.com/

Lardinois, F. (2018, October 6). MongoDB switches up its open source license.

https://techcrunch.com/2018/10/16/mongodb-switches-up-its-open-source-license/

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

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

reduxjs. (2016, April 22). *Reduxjs/redux.* Retrieved from

spring-projects (2019, March 26) *spring-projects/spring-boot.*

Retrieved from https://github.com/spring-projects/spring-boot/blob/master/LICENSE.txt

Appendix B – Copyright Compliance

All technologies that will be used to build Sententia are open source and available for community use. The table below includes a list of technologies, open-source licenses, and rationales for use.

|  |  |  |
| --- | --- | --- |
| **External Tool** | Open Source License | Rationale |
| Spring Boot | MIT (spring-projects, 2019) | Spring Boot has excellent support for microservices and MongoDB |
| React | Apache License 2.0 (Facebook, 2018) | React is an industry leading frontend framework. |
| Redux | MIT (reduxjs, 2016) | Redux is a state container that helps React store necessary data |
| Material-UI | MIT | Material-UI is a popular and industry-standard frontend library |
| MongoDB | SSPL (Lardinois, 2018) | MongoDB is the leading NoSQL database technology. |
| Docker | Apache 2.0 (Docker, 2020) | Docker is a leading virtualization technology and will allow each microservice to be deployed in its own container for scalability. |