**CST-323 Design Report - Milestone 3: Cloud Test Application on Cloud**

|  |  |  |
| --- | --- | --- |
| **Topic:** | *Milestone 3: Cloud Test Application on Cloud* | |
| **Date:** | *03/10/2018* | |
| **Revision:** | *3.1* | |
| **Team:** | 1. *Connor Low* | |
| 1. *Trevor Moore* | |
| **Weekly Team Status Summary:** | |  |  | | --- | --- | | **User Story** | **Team Member** | | *Completed Design Report & Draft Diagrams* | *Connor & Trevor* | | *Completed Full Front End Interface* | *Connor* | | *Completed Full Back End API* | *Trevor* | | |
| **Project URLs:** | **Application GIT Repository:**  <https://github.com/ConnorJamesLow/cloud-client>  **Application (BETA Branch) GIT Repository:**  <https://github.com/TrevorMooreGCU/cloud-client/tree/BETA2>  **Backend API GIT Repository**  <https://github.com/ConnorJamesLow/the-cloudiest-repo>  **Screencast Link:**  <https://www.useloom.com/share/69b5708e7a1b44bcac22fa4f976b5666>  **Application:**  <https://gcu-cloud-chat.herokuapp.com/>  **Application (BETA Branch):**  <https://gcu-cloud.herokuapp.com/>  **Swagger API Documentation:**  <https://app.swaggerhub.com/apis-docs/TrevorGCU/gcu-cloud/1.0.0> | |
| **Peer Review:** | *Yes* | We acknowledge that our team has reviewed this report and we agree to the approach we are all taking. |

**Planning Documentation**

**Project Management Outline:**

*For this project, our team is using the Agile Scrum development methodology. We will be keeping track of our individual tasks by maintaining a product backlog, sprint backlog, and work burndown chart.*

*Our team will make daily updates to our progress in the work burndown chart. Our team will perform daily standups everyday (not literally, unfortunately). We will perform these via in person on Tuesdays/Thursday in class, and also every other day of the week via Discord direct messaging.*

*Best case scenario, we will push updates on code to source control (BitBucket repository) every 30 minutes of development, and if that cannot be met, a max of pushing every hour will suffice.*

*Our team will hold each other accountable to commenting practically every line of code (within good reason). If a team member does not comment, the other team member reserves the right to call them out on it and request that they go back and add comments to all the code that’s lacking.*

*Our team will have two different roles for dividing the workload each sprint, with team members generally swapping roles each sprint:*

1. *Team Member will handle majority of back-end work.*

*Example: database design, business logic, data access logic, and models*

1. *Team Member will handle all front-end work and the remaining back end work*

*Example: All pages/forms, css, and controllers*

*Workload resource assignment will most likely be on a case to case basis, and the team will do its best to assign work as evenly as possible, keeping in mind that each member needs to get a chance to learn everything.*

**Design Documentation**

**General Technical Approach:**

***Summary:***

*A web-based chat application that allows users to send and receive messages in real time.*

***Domain and Products:***

*The data managed will consist of user information and messages sent between users. Message data will be subject to all CRUD methods. Users will be able to visit our app and enter their nickname, which will be stored in a cookie. Once they have done so, they will be able to send messages to other users in a global forum (the nickname they entered will be displayed by messages they send). All users will be able to view all messages sent by all other users. All users will also be able to update and delete any message sent by any user.*

*Messages will be sent as plain text. Messages will be created in a simple ajax-enabled form consisting of a text area and a send button. Messages will be displayed in the global chat after sending. A continuous script will fetch check for new messages to simulate real-time delivery. Upon opening the global or a direct chat, previous messages will be loaded in blocks of 25.*

***Approach:***

*Our team’s approach to this project is to incorporate industry best practices to the entire scope of the project, including the software development aspect as well as the design documentation (Entity-Relationship diagrams, Unified Modeling Language diagrams, etc.) and development methodology (Agile Scrum). From this standpoint, we want to successfully include all business requirements into our project. If possible, we want to root out all bugs prior to each deployment. We will accomplish this by developing a testing plan for each iteration/sprint as we go, and we will execute each testing plan for each module at the end of every sprint, to ensure that we can identify any bugs and isolate them as best as possible to find a resolution.*

*Our team will use N-Layer architecture for this project, and we will implement multiple open source technologies such as Bootstrap and jQuery. We will host our application on Heroku and utilize the many tools that the cloud provides, which includes leveraging a MySQL relational database. We will create detailed documentation of our design and architecture and update these designs as we move along through each sprint, adjusting when and where we see fit. We will develop our design documents prior to coding, as this will ensure a smooth transition between design and code, and ultimately create a much faster delivery of each sprint. Our team will make good use of commenting code and will do so by implementing things such as JavaDoc.*

**Key Technical Design Decisions:**

***Technologies:***

* *Server*
  + *Language: Java (JDK 1.8.0)*
  + *Web-app framework: Spring (5.1.4)*
* *Front End*
  + *jQuery (3.3.1)*
  + *Bootstrap (4.2.1)*
* *Environment and tools*
  + *IDE: Eclipse (2018-12)*
  + *Version Control: git (2.16.1), GitHub*
  + *Hosting Solution and CLI: Heroku*

***Decisions:***

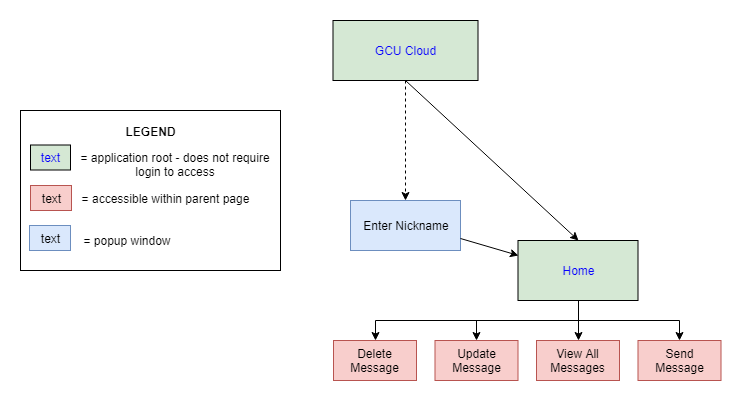
*Backend API will be a complete enterprise class N-Layer application and will have distinct and separate components, business services, and persistence services. Some of our team’s key technical design decisions are as follows:*

* ***FINAL CLOUD PROVIDER:*** *Heroku*
* *Application will be developed in Eclipse IDE using the Java programming language*
* *Spring MVC framework*
* *Apache Maven to build our project*
* *Spring Core for business services*
* *Spring JDBC for persistence*
* *jQuery for a user-friendly interface*
* *Bootstrap to make it a responsive application*
* *Tomcat for the Java application server*
* *MySQL for our relational database*
* *User-friendly welcome page/nickname prompt popup*
* *User-friendly landing page/main page for viewing, creating, updating, and deleting messages*
* *Application will perform data validation on all data entry fields*
* *Application will be responsive (for desktop and mobile use)*
* *Application will handle errors and exceptions using a global exception handler defined in the applicationConfiguration.xml*

**Risks:**

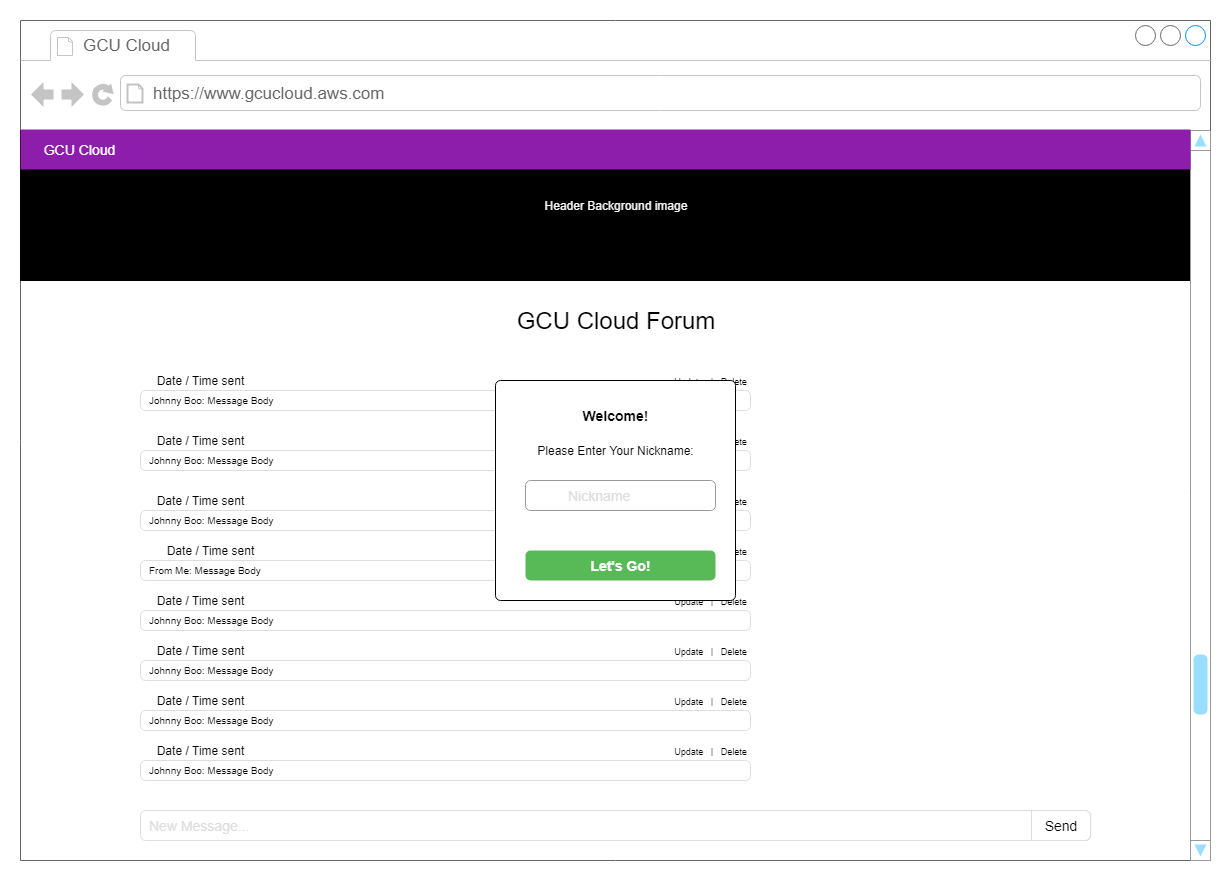
|  |  |  |
| --- | --- | --- |
| **Risk** | **Action Plan** | **Due Date** |
| *Failure by teammates to comment code; delayed delivery time because of reanalysis of code.* | *Nag each other to always comment as we go instead of coding and then going back to comment.* | 02/24/19 |
| *Failure by teammates to push/pull from source control/messing up project configuration as a result.* | *Always pull before a push, and have a running project separate from the project in the repo git folder.* | 01/27/19 |
| *Failure to successfully deploy our application to our desired hosting service.* | *Ensure at least one of us has an active subscription (whether free or paid, if need be) to our desired hosting service. Also ensure we will be able to deploy our desired stack (Spring) onto our desired service of choice prior to going too far down the rabbit hole.* | 01/27/19 |
| *Failure by teammates to update Work Breakdown Chart/communicate/do their work* | *Always text each other; be in (reasonably) constant communication on the status of the project. Do not be passive – communicate!* | 01/27/19 |

**Sitemap Diagram:**

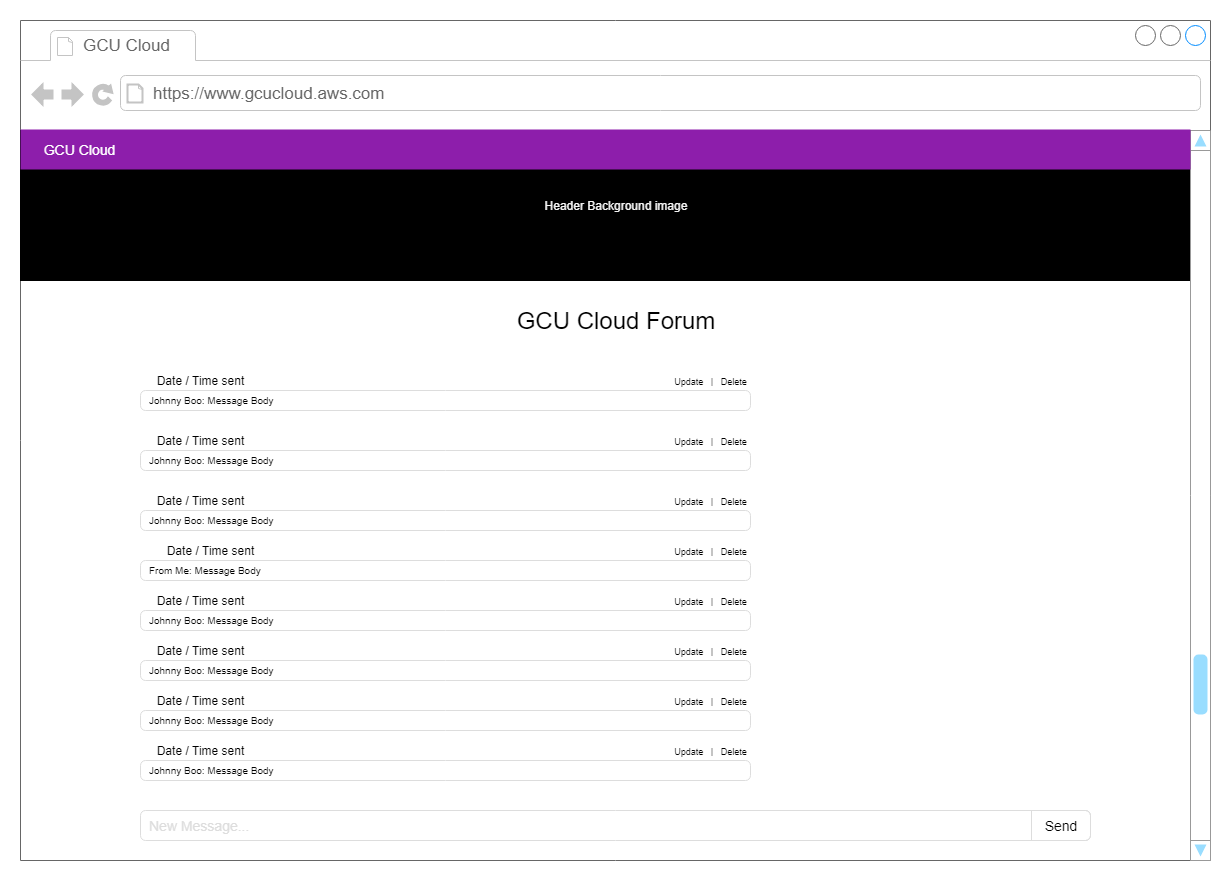
****

**User Interface Diagrams:**

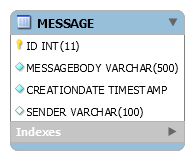
Welcome Page/Root of the application (notice the welcome popup with nickname prompt):

****

Home page:

****

**ER Diagram (MySQL):**



**DDL Script:**

CREATE TABLE `MESSAGE` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`MESSAGEBODY` varchar(500) COLLATE utf8\_unicode\_ci NOT NULL,

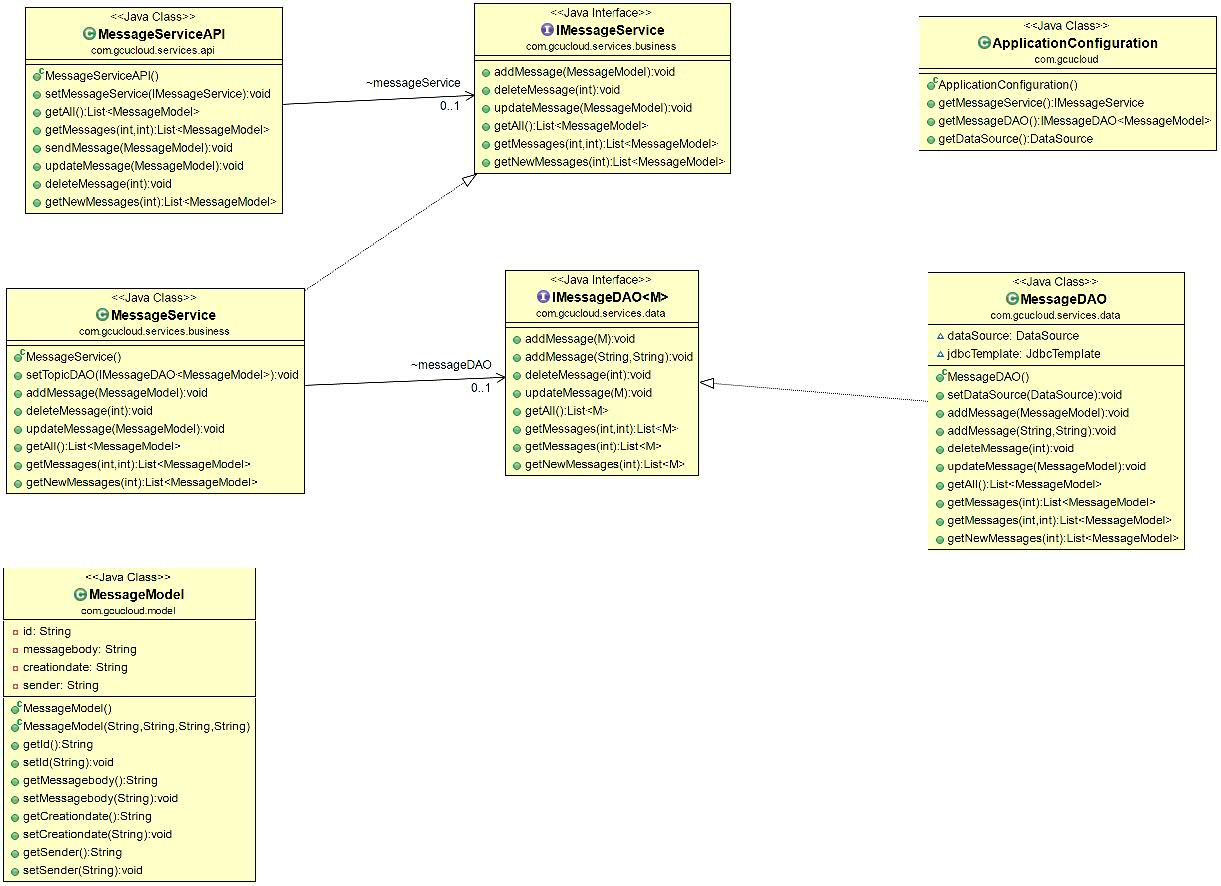
`CREATIONDATE` timestamp NOT NULL,

`SENDER` varchar(100) COLLATE utf8\_unicode\_ci DEFAULT 'Anonymous',

PRIMARY KEY (`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=21 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci

**UML Class Diagrams:**



**Application GIT Repository:**

[*https://github.com/ConnorJamesLow/cloud-client*](https://github.com/ConnorJamesLow/cloud-client)

**Application (BETA Branch) GIT Repository:**

<https://github.com/TrevorMooreGCU/cloud-client/tree/BETA2>

**Backend API GIT Repository**

[*https://github.com/ConnorJamesLow/the-cloudiest-repo*](https://github.com/ConnorJamesLow/the-cloudiest-repo)

**Screencast Link:**

[*https://www.useloom.com/share/69b5708e7a1b44bcac22fa4f976b5666*](https://www.useloom.com/share/69b5708e7a1b44bcac22fa4f976b5666)

**Heroku App Link:**

[*https://gcu-cloud-chat.herokuapp.com/*](https://gcu-cloud-chat.herokuapp.com/)

**Heroku App Link (BETA Branch):**

<https://gcu-cloud.herokuapp.com/>

**Swagger API Documentation:**

<https://app.swaggerhub.com/apis-docs/TrevorGCU/gcu-cloud/1.0.0>

**API Endpoints (Spring Rest):**

GET: [*https://gcucloud.herokuapp.com/messaging/getall*](https://gcucloud.herokuapp.com/messaging/getall)

GET: [*https://gcucloud.herokuapp.com/messaging/getmessages?index=0&count=5*](https://gcucloud.herokuapp.com/messaging/getmessages?index=0&count=5)

GET: [*https://gcucloud.herokuapp.com/messaging/get-new-messages?index=59*](https://gcucloud.herokuapp.com/messaging/get-new-messages?index=59)

DELETE: [*https://gcucloud.herokuapp.com/messaging/deletemessage/20*](https://gcucloud.herokuapp.com/messaging/deletemessage/20)

PUT: [*https://gcucloud.herokuapp.com/messaging/updatemessage*](https://gcucloud.herokuapp.com/messaging/updatemessage)

POST: [*https://gcucloud.herokuapp.com/messaging/sendmessage*](https://gcucloud.herokuapp.com/messaging/sendmessage)