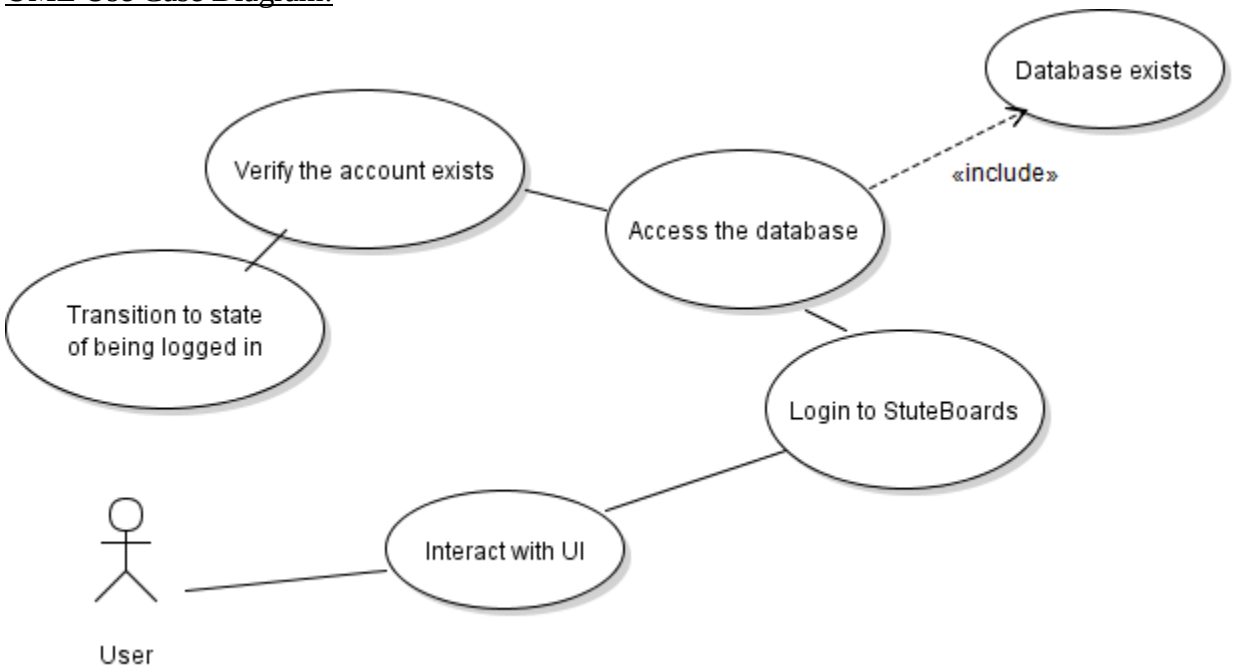
Assignment 2 --- Team Won

Feature: Log In

UML Use Case Diagram:



User Story and Test Story:

Title: User Login

Acceptance Test: usrLogin

Priority: 1

Story Points 14

As a user,

I want to be able to login to the site

so that I can browse posts and post on the site.

Behavior Test Template for User Login:

Given that a user has a valid StuteBoards account

and access to the Internet,

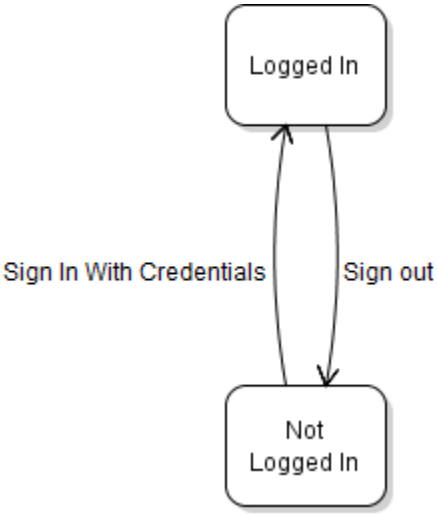
when the user attempts to login to the site

then the user gains access to the site

and can view posts on the site

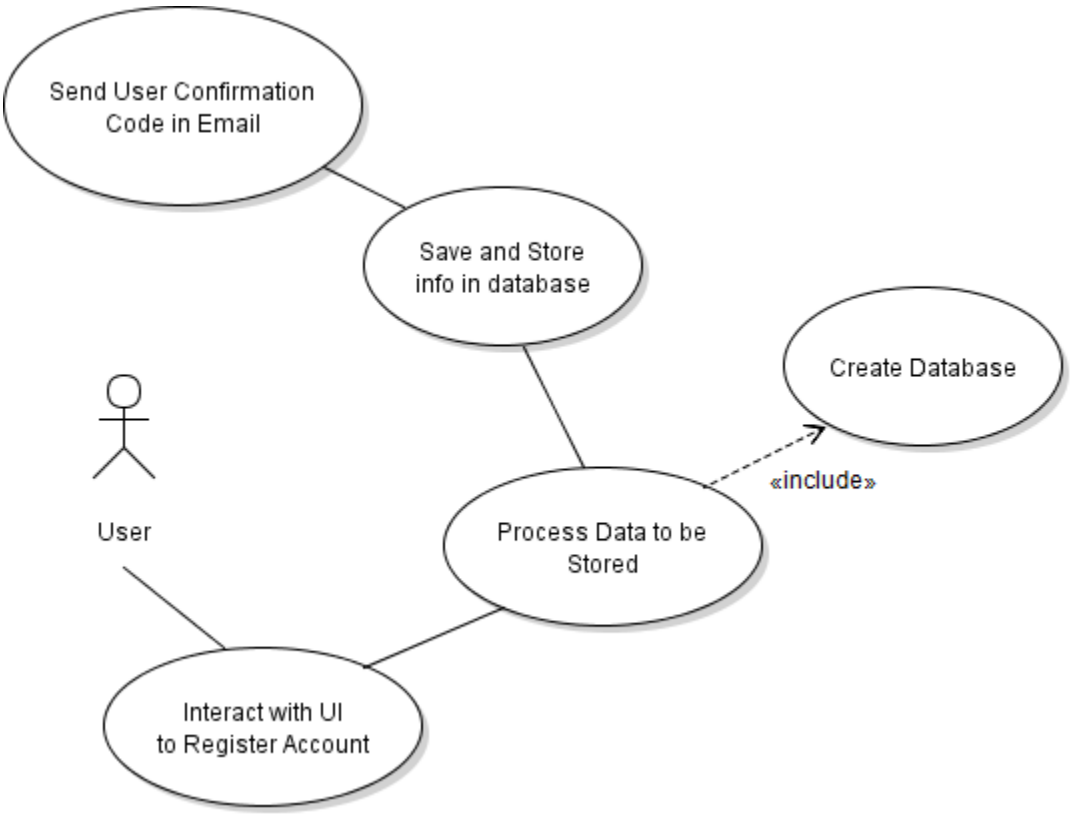
and can post on the site.

State Diagram:



Feature: User Registration

UML Use Case Diagram:



User Story and Test Story:

Title: User Registration

Acceptance Test: canRegister

Priority: 1

Story Points 14

As a user,

I want to be able to register for the site

so that I can potentially(must confirm email) browse posts and post on the site.

Behavior Test Template for User Registration:

Given that a user has a valid Stevens email

and access to the Internet,

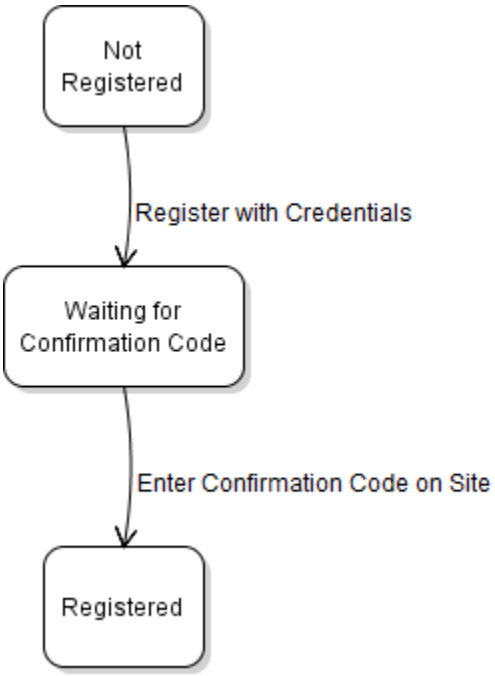
when the user attempts to register for the site

then the user creates an account for the site

and their information is secured

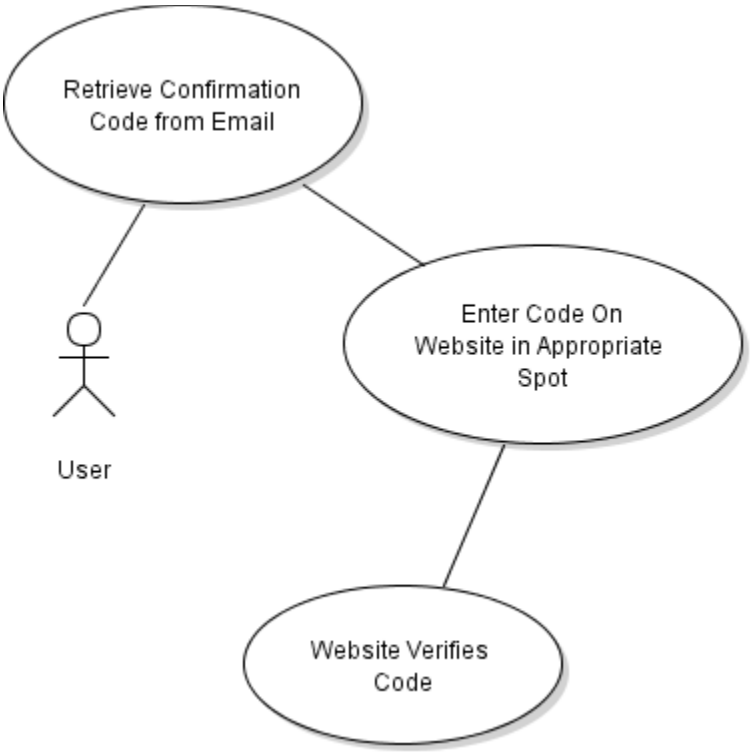
and they can confirm their account at a later time.

State Diagram:



Feature: User Confirmation

UML Use Case Diagram:



User Story and Test Story:

Title: User Confirmation

Acceptance Test: canConfirm

Priority: 1

Story Points 7

As a user,

I want to be able to confirm/validate my account on StuteBoards

so that I can browse posts and post on the site.

Behavior Test Template for User Confirmation:

Given that a user has registered their Stevens Account successfully

and access to the Internet,

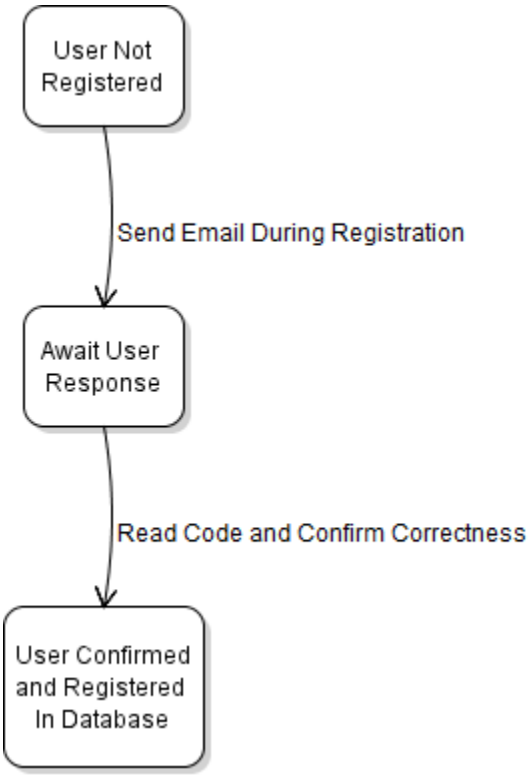
when the user attempts to confirm their email

then the user retrieves a confirmation code from their email

and they are granted access to the site and log into the site

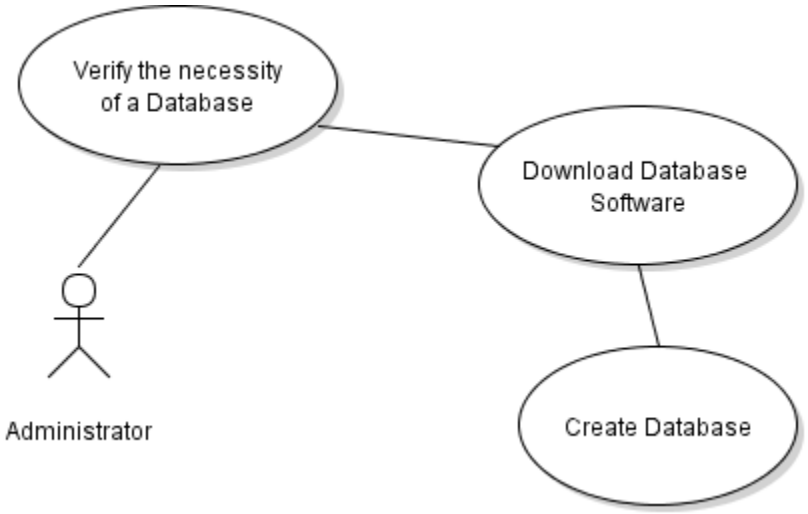
and they can post on threads.

State Diagram:



Feature: Create Database

UML Use Case Diagram:



User Story and Test Story:

Title: Create Database

Acceptance Test: createDB

Priority: 1

Story Points 4

As an administrator,

I want to be able to create a Database

so that users' information can be stored.

Behavior Test Template for Create Database:

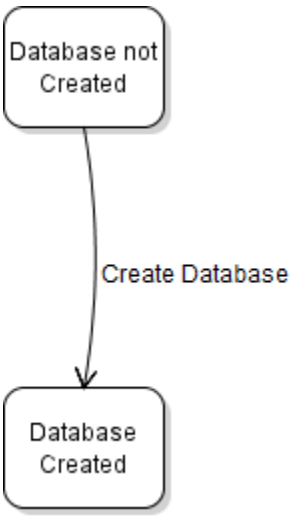
Given that an administrator has access to database software,

when there is no database

then the administrator will create a database

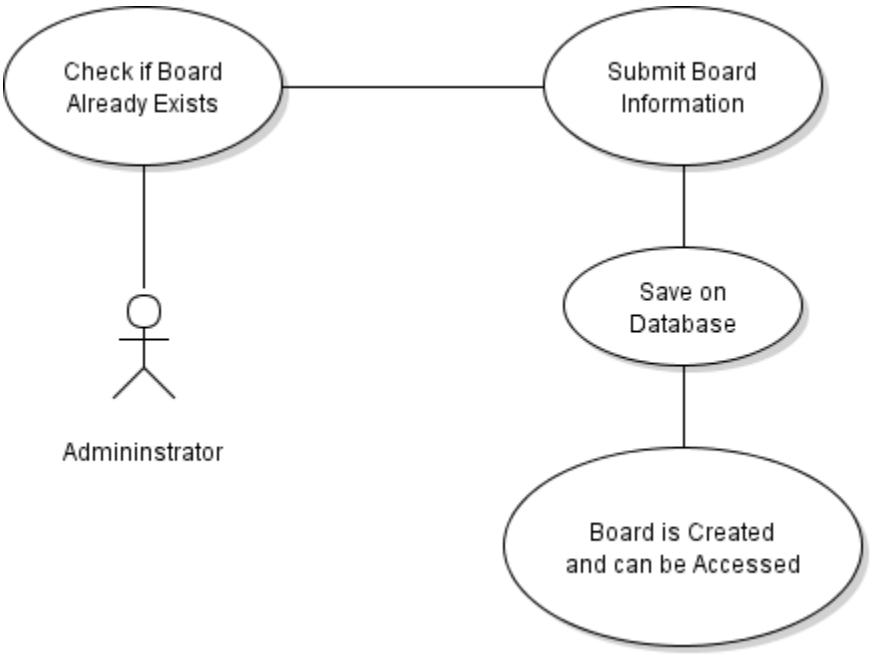
and give it the ability to store user and site information.

State Diagram:



Feature: Create Board

UML Use Case Diagram:



User Story and Test Story:

Title: Board Creation

Acceptance Test: createBoard

Priority: 1

Story Points 7

As an administrator,

I want to be able to create a new Message Board

so that there can be places to discuss a specific topic on the website for users.

Behavior Test Template for Board Creation:

Given that a user is an administrator

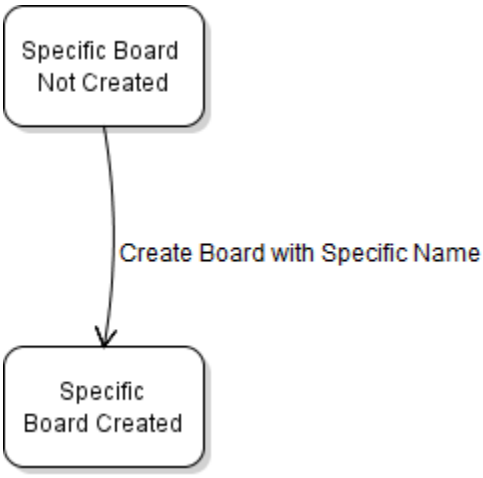
and the Board topic does not exists,

when the administrator wants to create a board

then the administrator can name the board

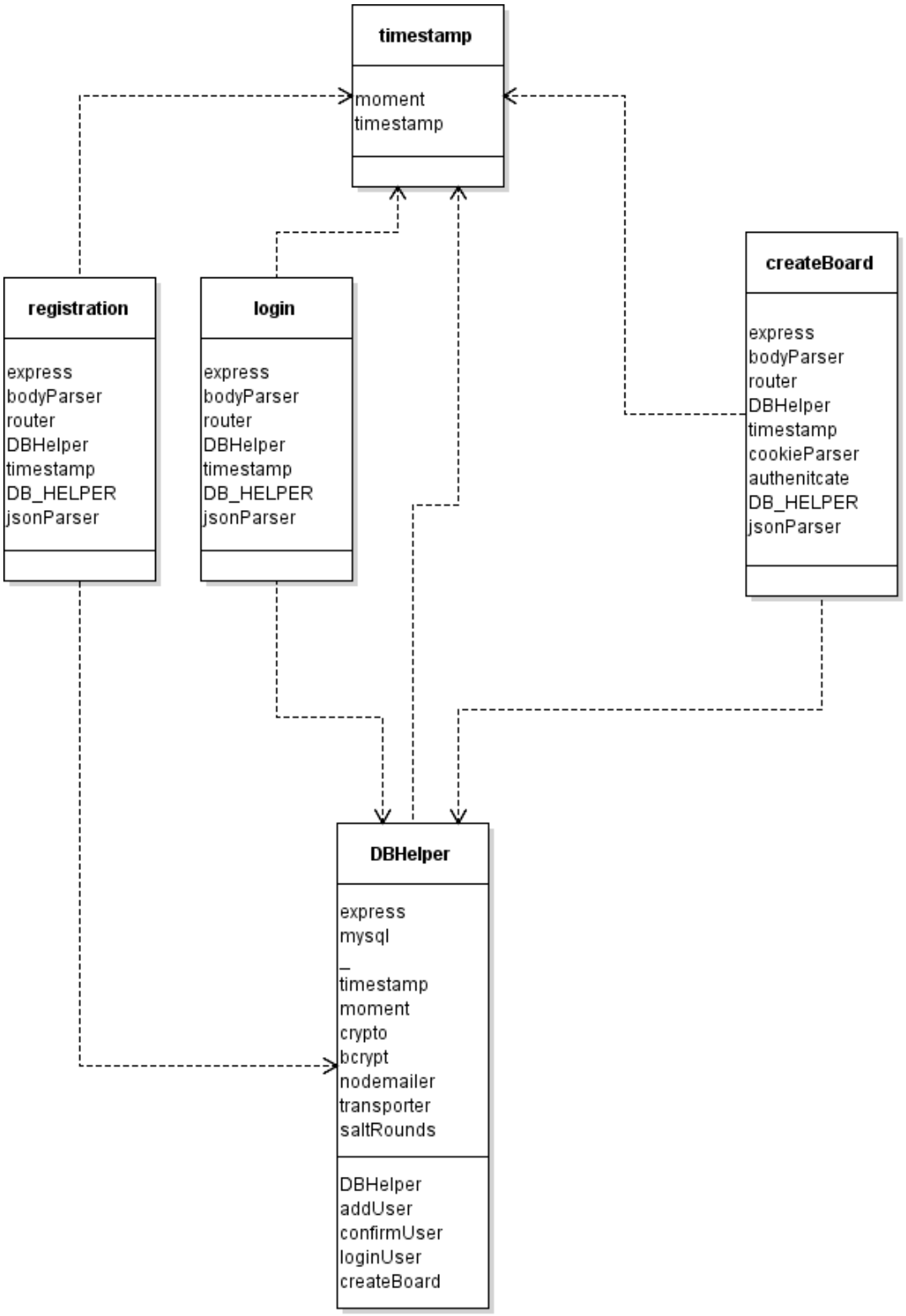
and make it available to all users.

State Diagram:



Class Diagram:

For our project, we did not use a typical object oriented approach. We used JavaScript, so we asked Professor Damopoulos how to structure our Class Diagram. He also said to explain that he gave us the ok to use this type of diagram:



High Level Design:

Our overall system is a website where users interact with a UI and the information taken at the front end is transferred and processed in the back end. At this point, our system is separated into subsystems: logging in, user registration, and board creation. All of the back end functionality is handled by functions defined in the helper file DBHelper.

Logging in: This subsystem first requires user input at the front end. The information is processed at the back end and the database is checked to see if the credentials given are registered and confirmed. If they are, the user is logged in, if not, an error message is sent.

User Registration: This subsystem first requires user input at the front end. The information is processed at the back end and it is stored in the database. A confirmation code is sent to the user's inputted email address. The user can then input the confirmation code and their account can be confirmed in the database.

Board Creation: This subsystem first requires user input at the front end. The information is processed at the back end and it is stored in the database. A new board is created that can be interacted with by users. If the board a user tries to create already exists, an error message is displayed.

All the data is stored in a MySQL database. If a user tries to view a board, the Board Creation and Logging in subsystems will interact. A user needs to be logged in before they can use or create a board. Also with User Registration and Logging in. A user has to be registered before they can log in.

Low Level Design:

We are not taking a typical object oriented design in this project, so we are using JavaScript to implement our features.

Logging in: This subsystem makes a call to the loginUser function of DBHelper. In DBHelper, it checks to make sure the user is in the database and valid. If the database cannot be reached, an error message reports the connection problem. Next, it makes sure the inputted credentials match the stored credentials in the database. If any of these are not valid, an error message is sent to the user telling them that their username or password is incorrect. When the credentials are accepted, an authentication token is generated that can be used for the user to navigate through the site.

User Registration: This subsystem makes use of the DBHelper functions addUser and confirmUser. The first call is to addUser, which takes the email address that the user is trying to register with and the password they want to use. If either of these things is undefined, an error will be displayed telling them this. If the email address does not come from the stevens.edu domain an error will be displayed telling them that the address is not valid. If these two conditions are met, then the function will attempt add the user info to the database. If that email address is already in the database, then the user will not be added and an error will tell them that the email is already in use. Once all of these requirements are confirmed, it generates a random 8-byte confirmation code, adds the user to the database, emails the confirmation code to the email the user submitted, and tells them to enter the code they receive on the confirmation page.

Once all of this is done, the program will make a call to confirmUser. confirmUser takes an email address and the confirmation code as parameters and preforms the same basic checks on the email to make sure it is valid. If the email is not already registered, an error message will tell them they are unregistered, and if the user is already confirmed an error message will prompt them to log in instead. After all of these checks, the confirmation code is verified. If it is invalid, an error message will inform the user and if it is valid, an authentication token is generated for the user. The authentication token is then added as cookie and is used to authenticate a user when they navigate through the site.

Board Creation: This subsystem makes a call to the createBoard function of DBHelper. In DBHelper, it checks to make sure the board does not exist yet. If it already exists, it sends an error message. If the board has not been created yet, it tries to store the information for the board in the database. If the database cannot be reached while creating the board, it sends an error message. If the board has not been created yet and it can connect to the database, a board is created of the inputted name.