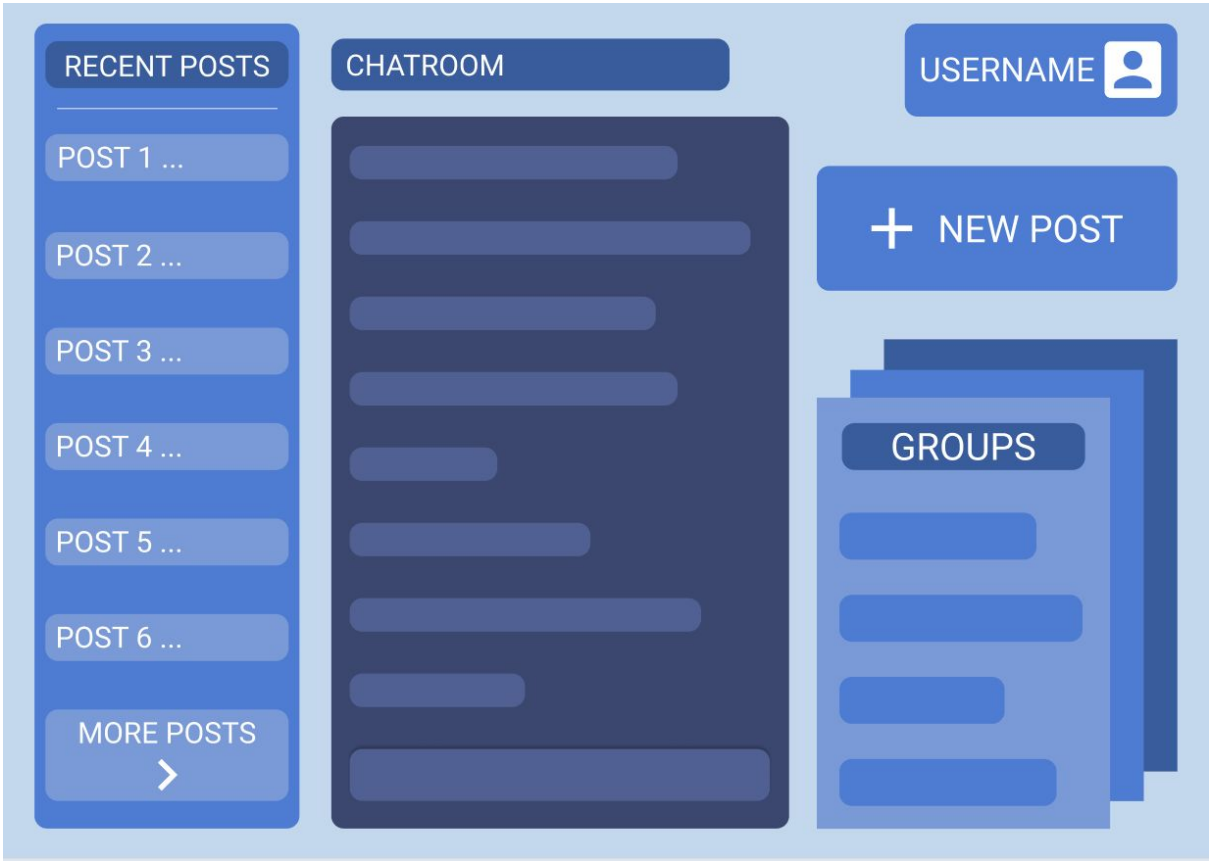
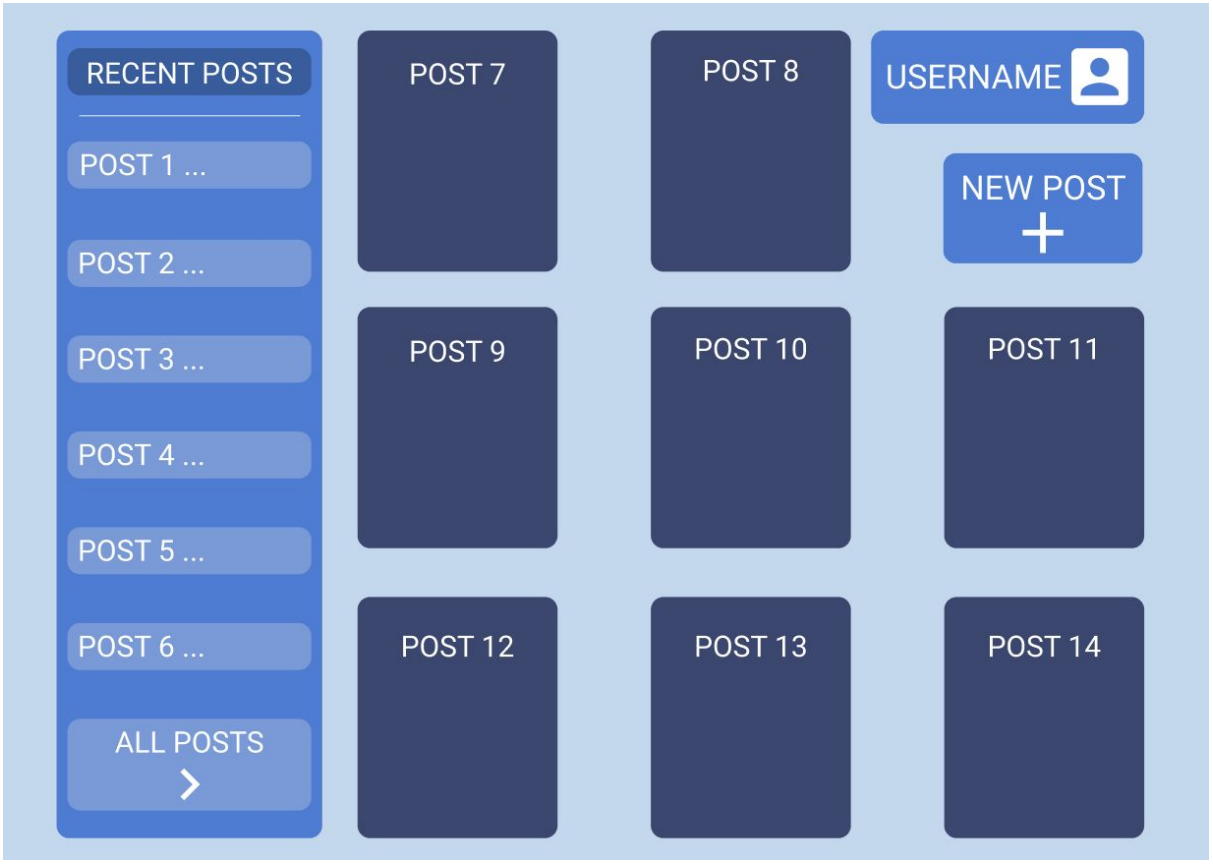
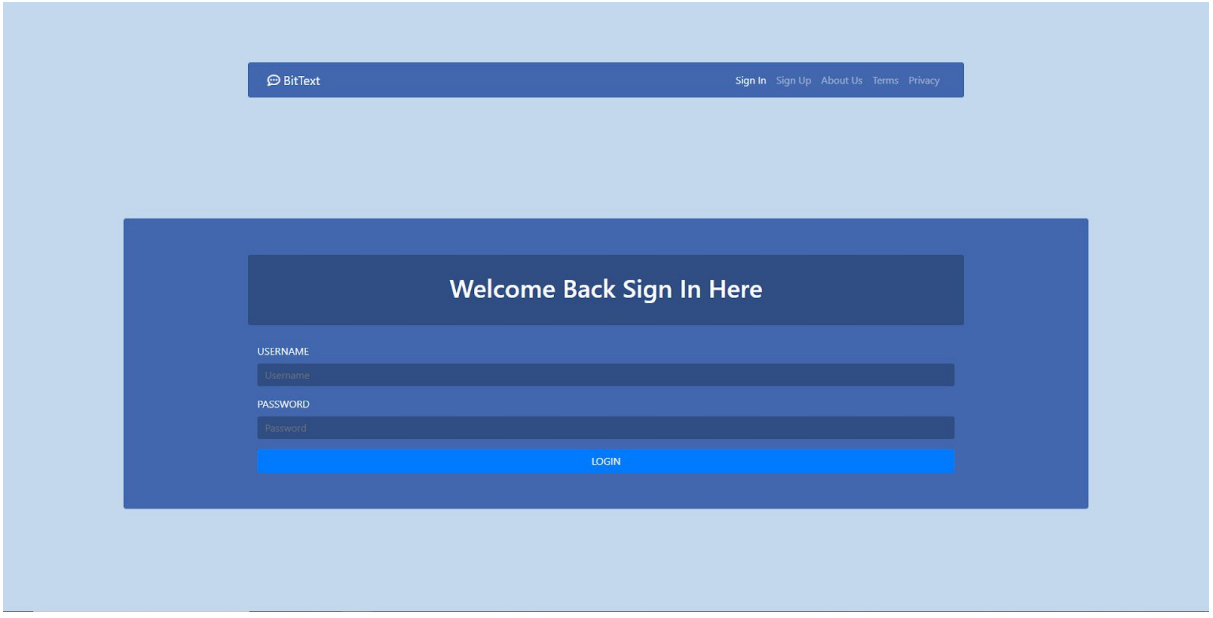
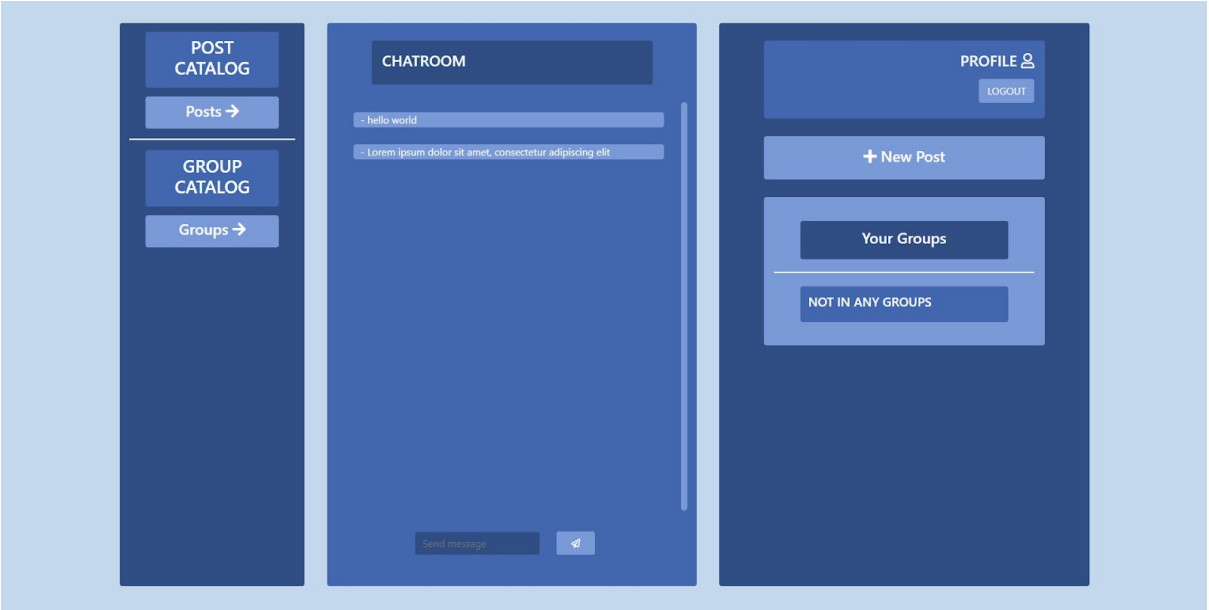
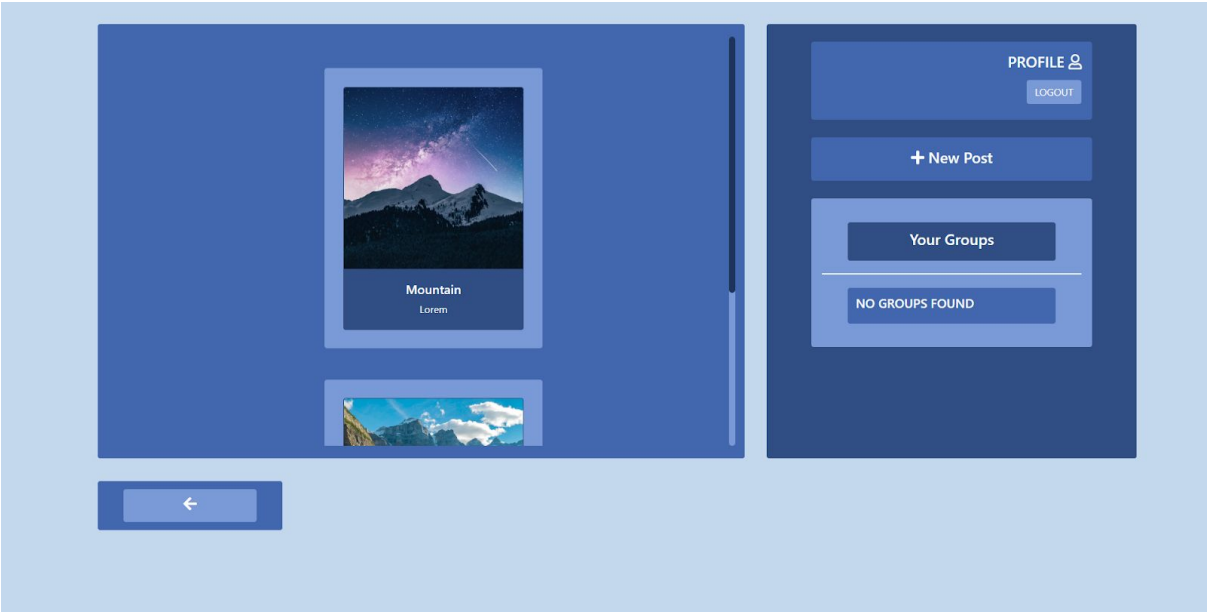
Comparing Wireframes and site -




RECENT POSTS
POST 1 ...
POST 2 ...
POST 3 ...
POST 4 ...
POST 5 ...
POST 6 ...
ALL POSTS
POST 7
POST 8
USERNAME
NEW POST
POST 9
POST 10
POST 11
POST 12
POST 13
POST 14


BitText
Sign In
Sign Up
About Us
Terms
Privacy
Welcome Back Sign In Here
USERNAME
Username
PASSWORD
Password
LOGIN





For the sign in page I changed the layout of both the logo and the sign in /sign up buttons and the info buttons from being how they were laid out before to having them ill in a navbar at the top of the page. I did this because I felt that it made the site look cleaner and less cluttered on the login page as well as having a navbar made it simpler to add to both the sign in/sign up pages.

For the user home page I made some pretty big changes first off the left side bar that used to contain recent user posts was scrapped altogether as I felt it was unnecessary as a user could just go to the post page to see the whole catalog of posts there so I instead changed it to link to the post catalog page as well as the group catalog page. The rest of the changes on this page were all visual and was mainly me just changing everything to have a more square look to it as I was unsure on how to replicate the complicated design I made in assignment three in bootstrap.

Finally the posts page on this page I added the full right sidebar to the page from the user home page as I felt it would be better if users could access their groups as well as make a new post from anywhere on the site instead of having to go back to the user home page to do it. Next it the styling and layout of the posts on the page, I changed it from the grid design I made in assignment three as it felt a bit messy and randomly laid out there so instead I chose to have all the posts stacked vertically inside of bootstrap cards and then added a scroll function to the div they were in so they would not fill up the entire page and make it extremely long with singular posts.

Description of the business - The aim of this business is to provide customers with a messaging/post sharing service letting users message each other and make posts including images to the site for other users to see. The potential users/customers for this site would be people that are looking for a site that provides instant messaging between users while supporting image posting and user groups.

How to run the site - Open the project with this link https://goor.me/nHK7d when opened press alt + T to open a second terminal window. In that second window cd into the course_project folder and type “mongod” into the terminal and run. Next go to the original terminal window and cd into the course_project folder and type “node app.js” and run. Wait until the console prints “BitText running on port 3000…” then go to the project tab in the top left then to running url and port and run the site on port 3000 and click the link to be sent to the site.

Feature/Content Explanations - One of the first features that was added was user authentication this includes the user sign up and sign in pages. This was done using the npm package passport as well as a user model file and a function to add the the routes to check if a user was logged in before accessing the page.

```
function isLoggedIn(req, res, next){
    if(req.isAuthenticated()){
        return next();
    }
    res.redirect("/");
}
```

The login function to check if a user is logged in before sending then to the page.

```
var mongoose = require("mongoose"),
    passportLocalMongoose = require("passport-local-mongoose");

var UserSchema = new mongoose.Schema({
    username: String,
    password: String
});

UserSchema.plugin(passportLocalMongoose);

module.exports = mongoose.model("User", UserSchema);
```

Model for users that contains a username and password.

Next I added the messaging section of the site where users can send text to be seen by all other users. This was done using a message model, ejs loop to print out all the messages in the database and a app.post route to save the message from the form to the database.

```
var mongoose = require("mongoose");

var messageSchema = mongoose.Schema({
    text: String
});

module.exports = mongoose.model("Message", messageSchema);
```

The message model.

```
<div class="scroll">
<!--    Below are all the user generated messages    -->
<p><% messages.forEach(function(message){ %>
    <div class="boxBlueHighlight rounded message"><strong><%= message.author %></strong> - <%= message.text %></div>
<% }) %> </p>
</div>
```

Loop to print out all user messages in the database.

```
app.post("/userhome", function(req, res){
    Message.create(req.body.message, function(err, message){
        if(err){
            console.log(err);
        }
        else{
            message.save();
            res.redirect("/userhome");
        }
    });
});
```

App.post route to save messages to the database from the form.

After that I made the posts section of the site using a posts model, a separate page for the form to make the post and a app.get to display the posts already made to the site and a app.post to send the data from the form into the database to be saved.

```
var mongoose = require("mongoose");

var PostSchema = new mongoose.Schema({
    name: String,
    image: String,
    description: String
});

module.exports = mongoose.model("Post", PostSchema);
```

Posts model containing a post name, image and description.

```
app.get("/posts", isLoggedIn, function(req, res){
    Post.find({}, function(err, allPosts){
        if(err){
            console.log(err);
        }
        else{
            res.render("posts", {posts:allPosts});
        }
    })
});

app.post("/posts", isLoggedIn, function(req, res){
    var name = req.body.name
    var image = req.body.image
    var description = req.body.description
    var newPost = {name:name, image:image, description:description}
    Post.create(newPost, function(err, post){
        if(err){
            console.log(err);
        }
        else{
            res.redirect("/posts");
        }
    });
});

app.get("/posts/new", isLoggedIn, function(req, res){
    res.render("new");
});
```

The routes for the post section of the site. One to display posts, another to save new posts to the database and the last to show the new post form.