

Project ON

Social networking platform

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND
ENGINEERING

NAME OF THE STUDENT: Jay Pathak

REGISTRATION NUMBER: 11702074

ROLL NUMBER: 13

SECTION: KM013



School of Computer Science and Engineering

Lovely Professional University Phagwara ,
Punjab (India)

GITHUB LINK: <https://github.com/jaypathak14/social-networking-site>

Repository is divided into three main packages:

- **Api** This package contains API for Social Networking App, built with Nodejs, Express, GraphQL, Apollo Server and MongoDB with Mongoose.
- **Frontend** Is a frontend for Social Networking App, built with React, GraphQL, Apollo Client and Styled Components.
- **Lib** Is a Nodejs command line script, that helps users to install the Social Networking App with one command.

Features

- **Messenger** Real time messaging system.
- **Notifications** Get instant notification when someone follows/messages you or likes/comments on your post.
- **User Status** Check if user is Online or not in real time.
- **News Feed** Fresh posts from people you are following.
- **Explore** New Posts and People.
- **Follow** a particular user and get notified for their activity.
- **Personalize Profile** With profile/cover photo and personal posts.
- **Authentication & Authorization** with Password reset functionality.

Ma'am tried to include comments as much as possible and Ma'am my postman was not working properly.

HOME

World Explorer

Search people

Dimi Mikadze

Home

Explore

People

Notifications


Messages

Pako Kochi

38 days

...

The clouds roll across the island covering all except the peaks that emerge. When they clear the ocean, the Gili Islands and Bali are all visible, but as fast as they come - they're gone again.



2 likes

2 comments

Like

Comment

Phillip Hodge

...

Suggestions For You

Quentin Brown

@quentin

William Wilkinson

@william

Paul Lee

@paul

Jeff Miranda

@jeff

Victor Bailey

@victor

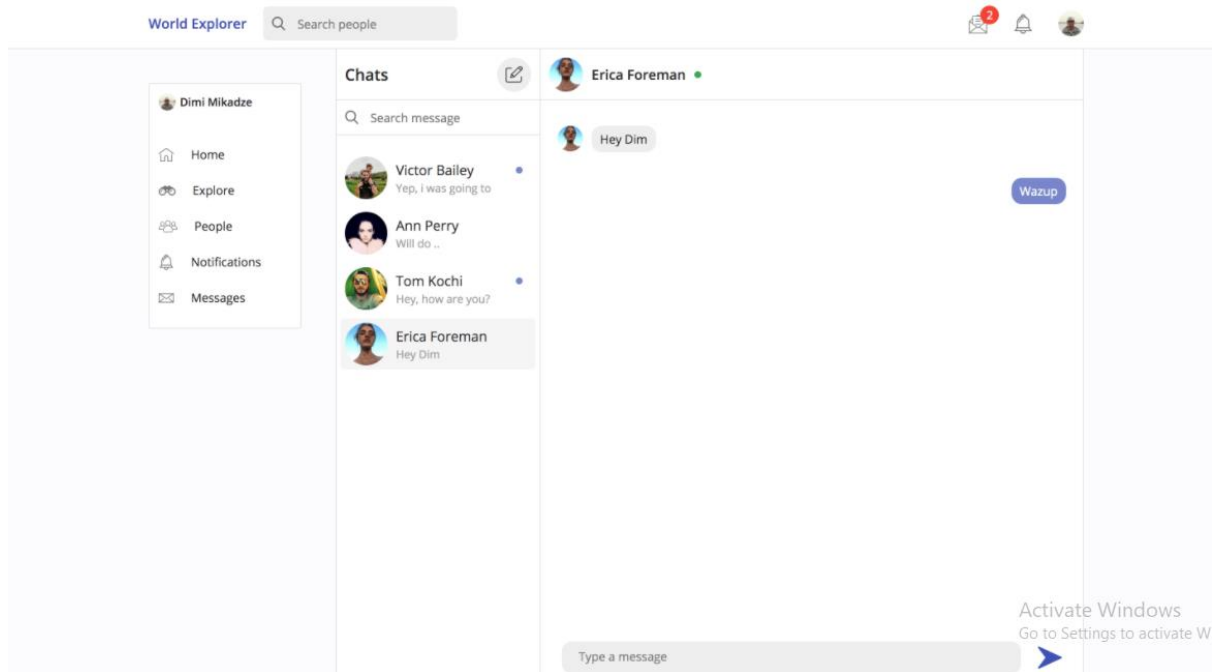
Tom Kochi

@tom

Activate Windows

Go to Settings to activate Wi

MESSAGES



PROFILE

World Explorer

Search people

Dimi Mikadze

HomeExplorePeopleNotificationsMessages

Dimi Mikadze

4 posts2 followers6 following

Add a post

Dimi Mikadze

38 days




Rickenbacker Causeway, United States


Activate Windows
Go to Settings to activate Win

PEOPLE


World Explorer


Search people








Dimi Mikadze


 Home

 Explore

 People


 Notifications

 Messages




Tom Kochi
@tom

Follow




Victor Bailey
@victor

Follow




Jeff Miranda
@jeff

Follow




Paul Lee
@paul

Follow




William Wilkinson
@william

Follow




Quentin Brown
@quentin

Follow



Joseph Gladden
@joseph

Follow



Louis Wakefield
@louis

Follow

Activate Windows

Go to Settings to activate Windows

EXPLORE

World Explorer

Search people

Dimi Mikadze


Home


Explore


People


Notifications


Messages














Activate Windows
Go to Settings to activate Win

NOTIFICATIONS

World Explorer

Search people

Dimi Mikadze

Home


Explore

People


Notifications

Messages


Paul Lee commented on your photo




Paul Lee commented on your photo




Paul Lee commented on your photo




Paul Lee likes your photo




Paul Lee likes your photo




Paul Lee likes your photo




Paul Lee likes your photo




Paul Lee started following you




Ann Perry commented on your photo



Ann Perry commented on your photo



Ann Perry commented on your photo



Activate Windows
Go to Settings to activate Win

CODE SNIPPETS

API MODELS

Server.js

```
mongoose
  .connect(process.env.MONGO_URL, {
    useCreateIndex: true,
    useNewUrlParser: true,
    useFindAndModify: false,
    useUnifiedTopology: true,
  })
  .then(() => console.log('DB connected'))
  .catch((err) => console.error(err));

// Initializes application
const app = express();

// Enable cors
const corsOptions = {
  origin: process.env.FRONTEND_URL,
  credentials: true,
};
app.use(cors(corsOptions));

// Create a Apollo Server
const server = createApolloServer(schema, resolvers, models);
server.applyMiddleware({ app, path: '/graphql' });

// Create http server and add subscriptions to it
const httpServer = createServer(app);
server.installSubscriptionHandlers(httpServer);

// Listen to HTTP and WebSocket server
const PORT = process.env.PORT || process.env.API_PORT;
httpServer.listen({ port: PORT }, () => {
  console.log(`server ready at http://localhost:${PORT}${server.graphqlPath}`)
;
  console.log(`Subscriptions ready at ws://localhost:${PORT}${server.subscriptionsPath}`);
});
```

Follow.js

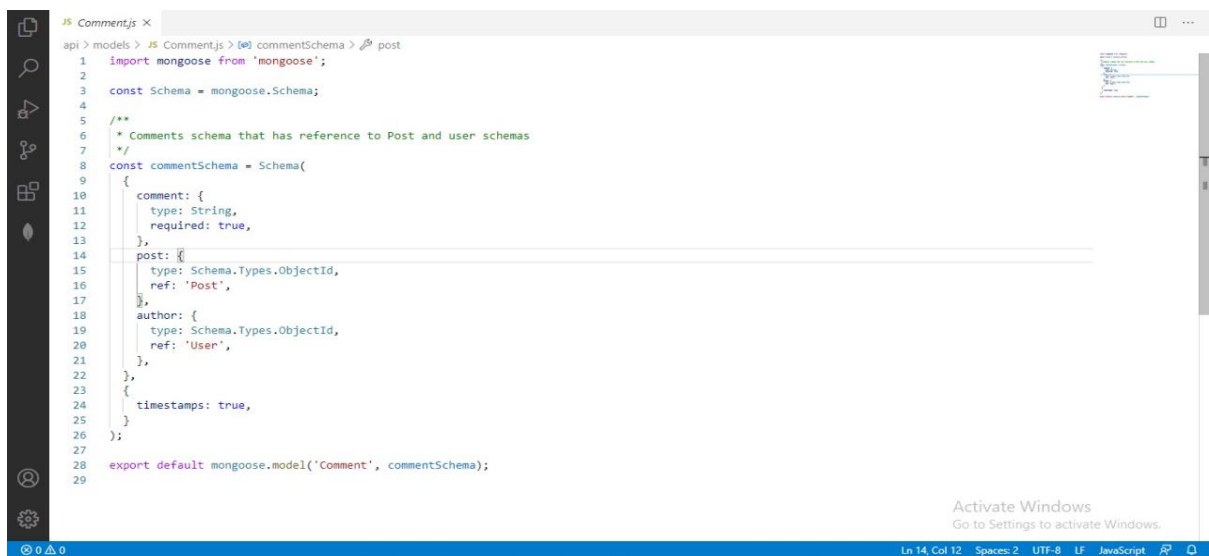


The screenshot shows a VS Code editor window with a file named 'Follow.js'. The code defines a Mongoose schema for a 'Follow' model. It includes imports for Mongoose, a schema definition with 'user' and 'follower' fields (both of type ObjectId and referencing 'User'), and timestamps. The schema is then registered with Mongoose. The status bar at the bottom indicates the current position is Line 1, Column 1.

```
1 import mongoose from 'mongoose';
2
3 const Schema = mongoose.Schema;
4
5 /**
6  * Follow schema that has references to User schema
7  */
8 const followSchema = Schema({
9   user: {
10     type: Schema.Types.ObjectId,
11     ref: 'User',
12   },
13   follower: {
14     type: Schema.Types.ObjectId,
15     ref: 'User',
16   },
17 }, {
18   timestamps: true,
19 });
20
21 export default mongoose.model('Follow', followSchema);
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript

Comment.js

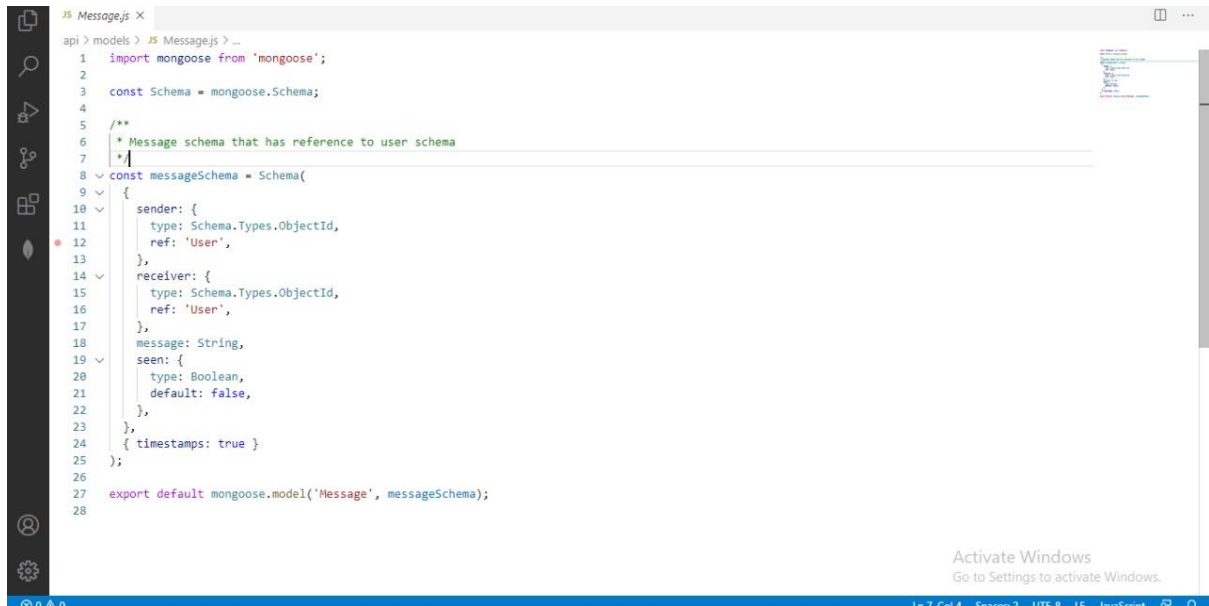


The screenshot shows a VS Code editor window with a file named 'Comment.js'. The code defines a Mongoose schema for a 'Comment' model. It includes imports for Mongoose, a schema definition with 'comment' (String), 'post' (ObjectId referencing 'Post'), and 'author' (ObjectId referencing 'User') fields, along with timestamps. The schema is then registered with Mongoose. The status bar at the bottom indicates the current position is Line 14, Column 12.

```
1 import mongoose from 'mongoose';
2
3 const Schema = mongoose.Schema;
4
5 /**
6  * Comments schema that has reference to Post and user schemas
7  */
8 const commentSchema = Schema({
9   comment: {
10     type: String,
11     required: true,
12   },
13   post: {
14     type: Schema.Types.ObjectId,
15     ref: 'Post',
16   },
17   author: {
18     type: Schema.Types.ObjectId,
19     ref: 'User',
20   },
21 }, {
22   timestamps: true,
23 });
24
25 export default mongoose.model('Comment', commentSchema);
```

Ln 14, Col 12 Spaces: 2 UTF-8 LF JavaScript

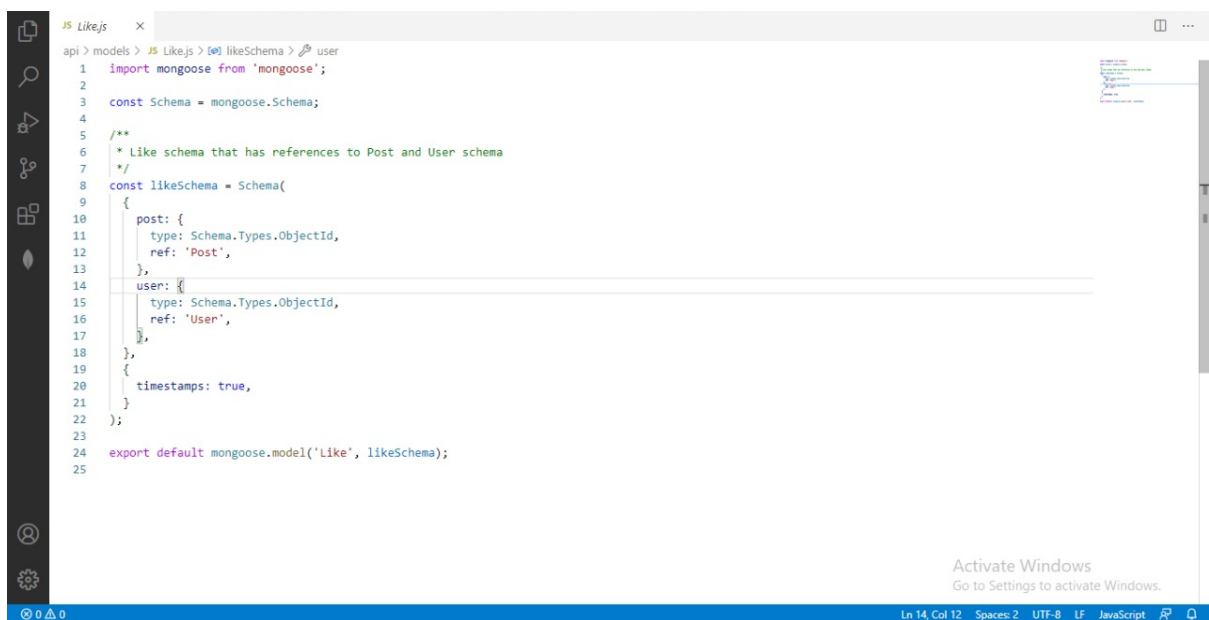
Message.js



```
1 import mongoose from 'mongoose';
2
3 const Schema = mongoose.Schema;
4
5 /**
6  * Message schema that has reference to user schema
7  */
8 const messageSchema = Schema({
9   sender: {
10     type: Schema.Types.ObjectId,
11     ref: 'User',
12   },
13   receiver: {
14     type: Schema.Types.ObjectId,
15     ref: 'User',
16   },
17   message: String,
18   seen: {
19     type: Boolean,
20     default: false,
21   },
22   timestamps: true
23 });
24
25 export default mongoose.model('Message', messageSchema);
```

Activate Windows
Go to Settings to activate Windows.

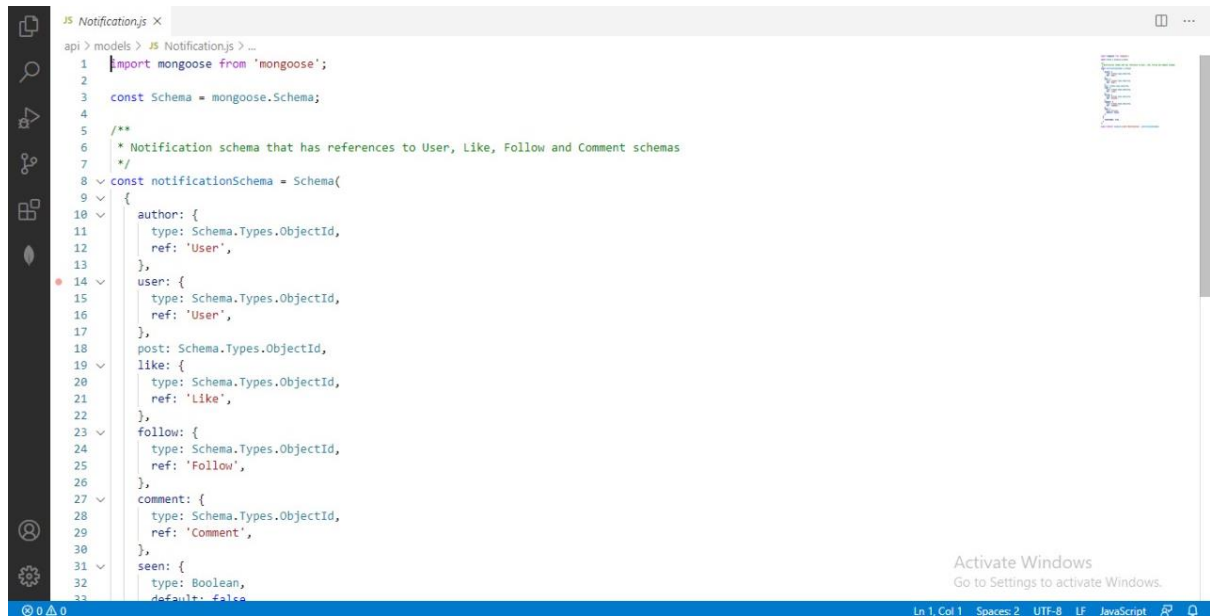
Like.js



```
1 import mongoose from 'mongoose';
2
3 const Schema = mongoose.Schema;
4
5 /**
6  * Like schema that has references to Post and User schema
7  */
8 const likeSchema = Schema({
9   post: {
10     type: Schema.Types.ObjectId,
11     ref: 'Post',
12   },
13   user: {
14     type: Schema.Types.ObjectId,
15     ref: 'User',
16   },
17   timestamps: true
18 });
19
20 export default mongoose.model('Like', likeSchema);
```

Activate Windows
Go to Settings to activate Windows.

Notification.js

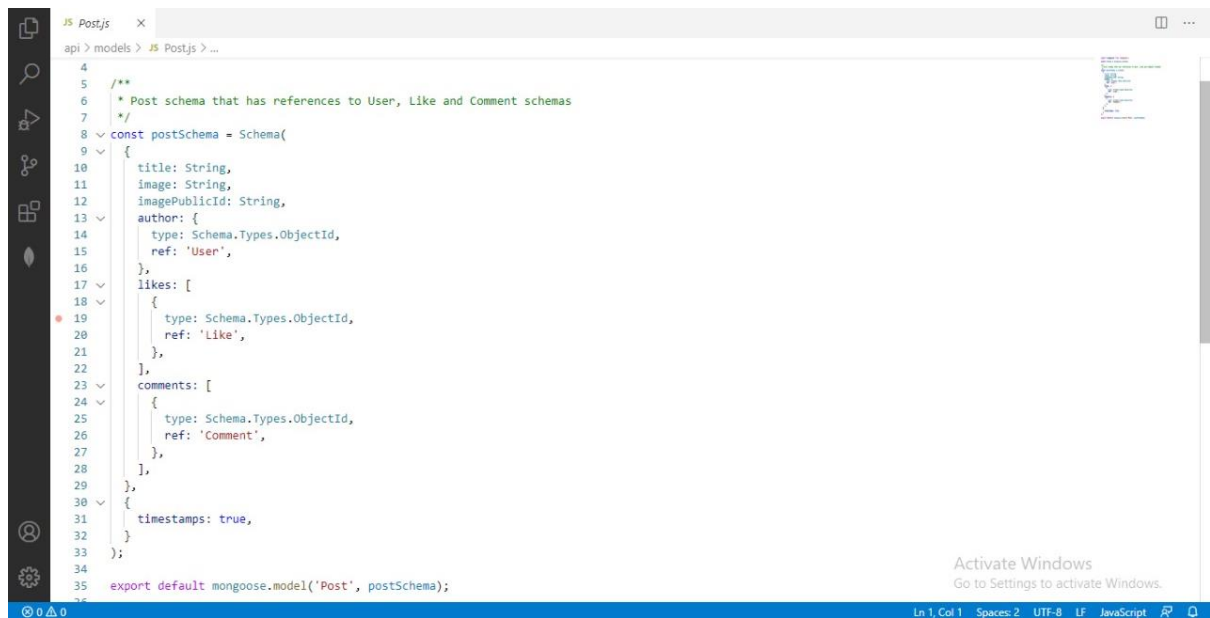


The screenshot shows a VS Code editor window with a file named 'Notification.js'. The code defines a Mongoose schema for notifications. It includes references to 'User', 'Like', 'Follow', and 'Comment' schemas. The schema has fields for 'author', 'user', 'post', 'like', 'follow', 'comment', and 'seen'.

```
1 import mongoose from 'mongoose';
2
3 const Schema = mongoose.Schema;
4
5 /**
6  * Notification schema that has references to User, Like, Follow and Comment schemas
7  */
8 const notificationSchema = Schema(
9   {
10     author: {
11       type: Schema.Types.ObjectId,
12       ref: 'User',
13     },
14     user: {
15       type: Schema.Types.ObjectId,
16       ref: 'User',
17     },
18     post: Schema.Types.ObjectId,
19     like: {
20       type: Schema.Types.ObjectId,
21       ref: 'Like',
22     },
23     follow: {
24       type: Schema.Types.ObjectId,
25       ref: 'Follow',
26     },
27     comment: {
28       type: Schema.Types.ObjectId,
29       ref: 'Comment',
30     },
31     seen: {
32       type: Boolean,
33       default: false,
34     },
35   }
36 );
37
38 export default mongoose.model('Notification', notificationSchema);
```

Activate Windows
Go to Settings to activate Windows.

Post.js



The screenshot shows a VS Code editor window with a file named 'Post.js'. The code defines a Mongoose schema for posts. It includes references to 'User', 'Like', and 'Comment' schemas. The schema has fields for 'title', 'image', 'imagePublicId', 'author', 'likes', 'comments', and 'timestamps'.

```
4
5 /**
6  * Post schema that has references to User, Like and Comment schemas
7  */
8 const postSchema = Schema(
9   {
10     title: String,
11     image: String,
12     imagePublicId: String,
13     author: {
14       type: Schema.Types.ObjectId,
15       ref: 'User',
16     },
17     likes: [
18       {
19         type: Schema.Types.ObjectId,
20         ref: 'Like',
21       },
22     ],
23     comments: [
24       {
25         type: Schema.Types.ObjectId,
26         ref: 'Comment',
27       },
28     ],
29     timestamps: true,
30   }
31 );
32
33 export default mongoose.model('Post', postSchema);
```

Activate Windows
Go to Settings to activate Windows.

FRONTEND

AuthHeader.js

```
const Root = styled.div`
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  display: flex;
  flex-direction: row;
  align-items: center;
  height: 80px;
  background-color: transparent;
`;

const StyledContainer = styled(Container)`
  width: 100%;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  padding: 0 ${p => p.theme.spacing.sm};

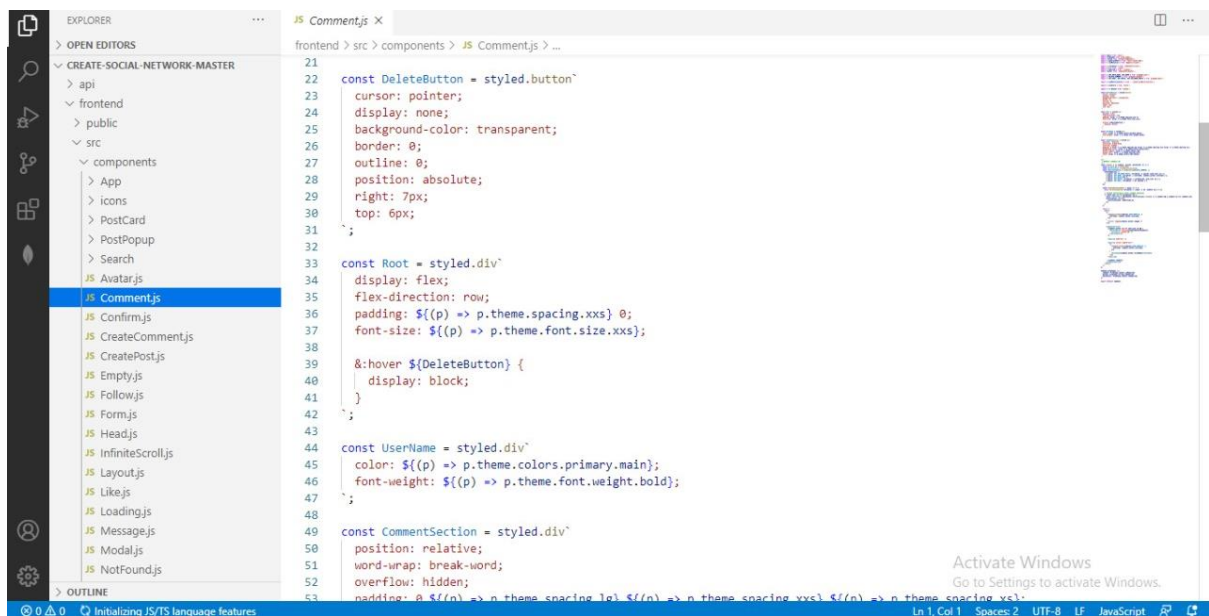
  @media (min-width: ${p => p.theme.screen.md}) {
    justify-content: space-between;
  }
`;

const Logo = styled(A)`
  display: none;
  color: ${p => p.theme.colors.white};
  font-size: ${p => p.theme.font.size.sm};
  font-weight: ${p => p.theme.font.weight.bold};

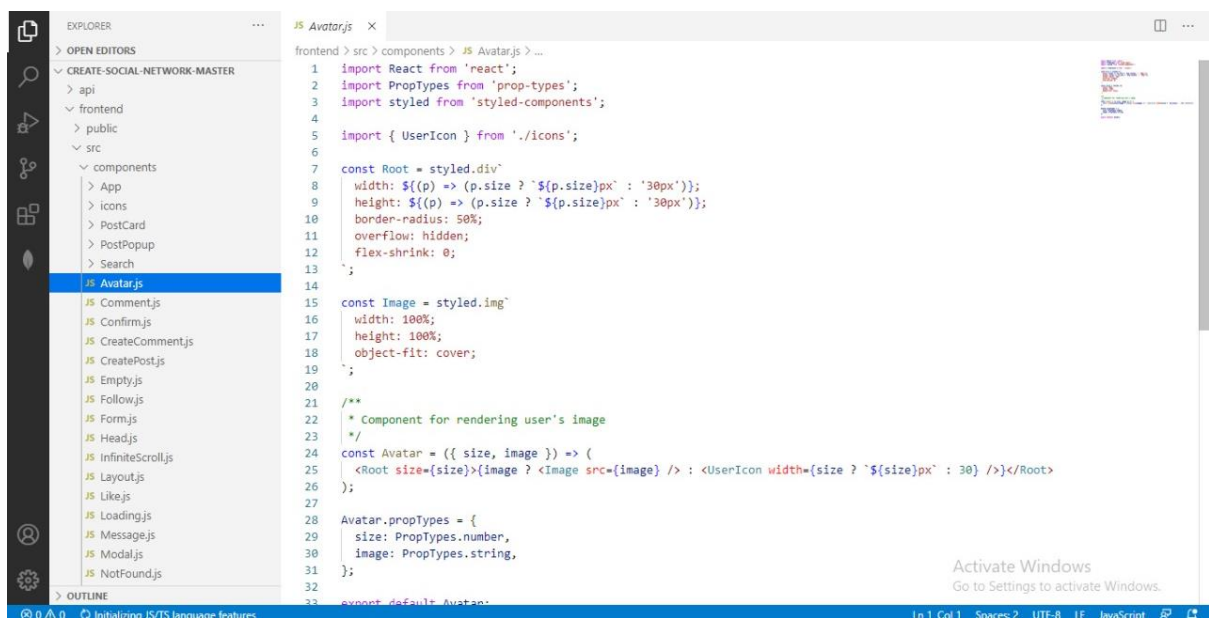
  &:hover {
    color: ${p => p.theme.colors.white};
  }

  @media (min-width: ${p => p.theme.screen.md}) {
    display: block;
  }
`;
```

Comment.js



Avatar.js




Follow.js

```
35 /**
36  * Component for rendering follow button
37  */
38 const Follow = ({ user }) => {
39   const [loading, setloading] = useState(false);
40   const [{ auth }] = useStore();
41   const notification = useNotifications();
42   const isFollowing = auth.user.following.find((f) => f.user === user.id);
43   // Detect which mutation to use
44   const operation = isFollowing ? 'delete' : 'create';
45   const options = {
46     create: {
47       mutation: CREATE_FOLLOW,
48       variables: { userId: user.id, followerId: auth.user.id },
49     },
50     delete: {
51       mutation: DELETE_FOLLOW,
52       variables: { id: isFollowing ? isFollowing.id : null },
53     },
54   };
55   const [mutate] = useMutation(options[operation].mutation, {
56     refetchQueries: [
57       { query: GET_AUTH_USER },
58       { query: GET_POSTS, variables: { authUserId: auth.user.id } },
59       {
60         query: GET_FOLLOWED_POSTS,
61         variables: {
62           userId: auth.user.id,
63           skip: 0,
64           limit: HOME_PAGE_POSTS_LIMIT,
65         },
66       },
67     ],
68   });
```

Message.js

```
19   opacity: 0;
20 }
21 to {
22   bottom: 0;
23   opacity: 1;
24 }
25 `;
26
27 /**
28  * Default styles for message
29  */
30 const Root = styled.div`
31   position: fixed;
32   width: 100%;
33   box-shadow: ${({p}) => p.theme.shadows.md};
34   padding: ${({p}) => p.theme.spacing.sm};
35   z-index: ${({p}) => p.theme.zIndex.xl};
36   display: flex;
37   flex-direction: row;
38   justify-content: center;
39   align-items: center;
40   background-color: ${({p}) => p.color && p.color};
41   animation: ${({fade}) 0.3s ease-out forwards};
42   color: ${({p}) => p.theme.colors.white};
43 `;
44
45 const Close = styled(Button)`
46   position: absolute;
47   right: 20px;
48   top: 24px;
49 `;
```

InfiniteScroll.js

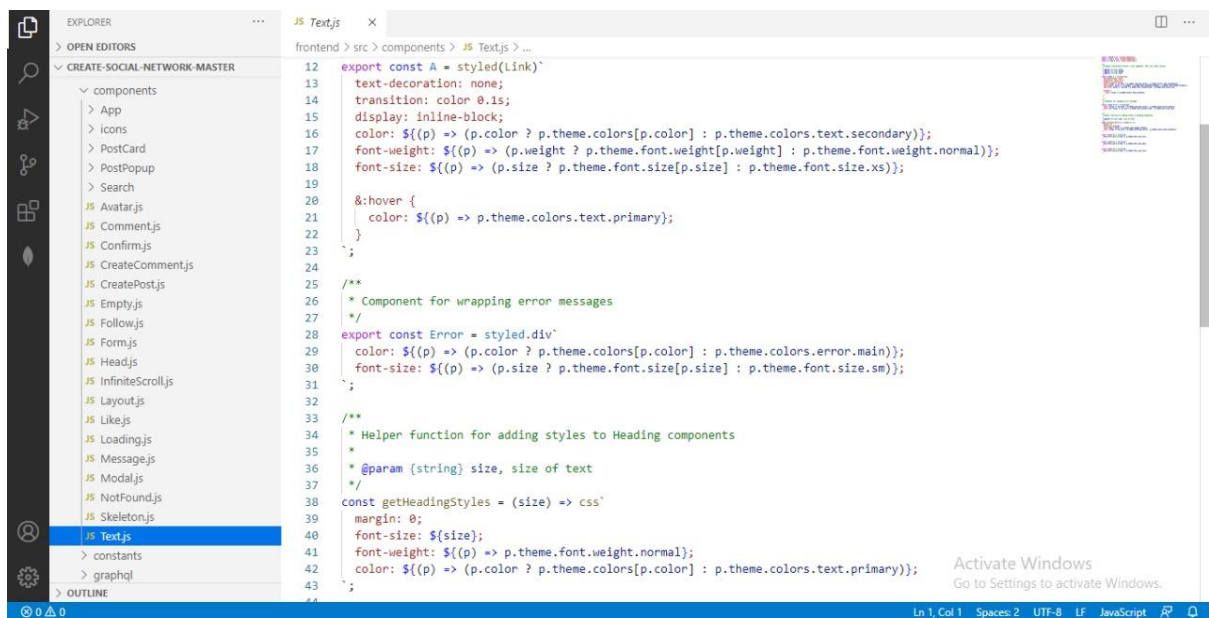


The screenshot shows the VS Code editor with the file `InfiniteScroll.js` open. The Explorer sidebar on the left shows a project structure for 'CREATE-SOCIAL-NETWORK-MASTER' with folders 'api', 'frontend', 'public', and 'src'. Under 'src', there is a 'components' folder containing various files, with `InfiniteScroll.js` selected. The main editor area shows the code for `InfiniteScroll.js`, which includes a `useEffect` hook to add and remove scroll event listeners, and a `loadMore` function. The code is as follows:

```
28 if (data.length >= count) {
29   window.removeEventListener('scroll', handleScroll);
30   return;
31 }
32
33 // Load more data if user has scrolled to bottom and if there's still data in db to display
34 if (scrolled) {
35   window.removeEventListener('scroll', handleScroll);
36   loadMore();
37 }
38 },
39 [count, data.length, dataKey, fetchMore, variables]
40 );
41
42 useEffect(() => {
43   window.addEventListener('scroll', handleScroll);
44
45   return () => window.removeEventListener('scroll', handleScroll);
46 }, [handleScroll]);
47
48 return children(data);
49 };
50
51 InfiniteScroll.propTypes = {
52   data: PropTypes.array.isRequired,
53   dataKey: PropTypes.string.isRequired,
54   count: PropTypes.number.isRequired,
55   fetchMore: PropTypes.func.isRequired,
56   variables: PropTypes.object.isRequired,
57   children: PropTypes.func.isRequired,
58 };
59
```

The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', and 'JavaScript'.

Text.js



The screenshot shows the VS Code editor with the file `Text.js` open. The Explorer sidebar on the left shows the same project structure as the previous screenshot, with `Text.js` selected under the 'components' folder. The main editor area shows the code for `Text.js`, which includes a `styled` function for Link components, a `styled.div` for Error components, and a `getHeadingStyles` function. The code is as follows:

```
12 export const A = styled(Link)`
13   text-decoration: none;
14   transition: color 0.1s;
15   display: inline-block;
16   color: ${(p) => (p.color ? p.theme.colors[p.color] : p.theme.colors.text.secondary)};
17   font-weight: ${(p) => (p.weight ? p.theme.font.weight[p.weight] : p.theme.font.weight.normal)};
18   font-size: ${(p) => (p.size ? p.theme.font.size[p.size] : p.theme.font.size.xs)};
19
20   &:hover {
21     color: ${(p) => p.theme.colors.text.primary};
22   }
23 `;
24
25 /**
26  * Component for wrapping error messages
27  */
28 export const Error = styled.div`
29   color: ${(p) => (p.color ? p.theme.colors[p.color] : p.theme.colors.error.main)};
30   font-size: ${(p) => (p.size ? p.theme.font.size[p.size] : p.theme.font.size.sm)};
31 `;
32
33 /**
34  * Helper function for adding styles to Heading components
35  *
36  * @param {string} size, size of text
37  */
38 const getHeadingStyles = (size) => css`
39   margin: 0;
40   font-size: ${size};
41   font-weight: ${(p) => p.theme.font.weight.normal};
42   color: ${(p) => (p.color ? p.theme.colors[p.color] : p.theme.colors.text.primary)};
43 `;
44
```

The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', and 'JavaScript'.

NODE FILES

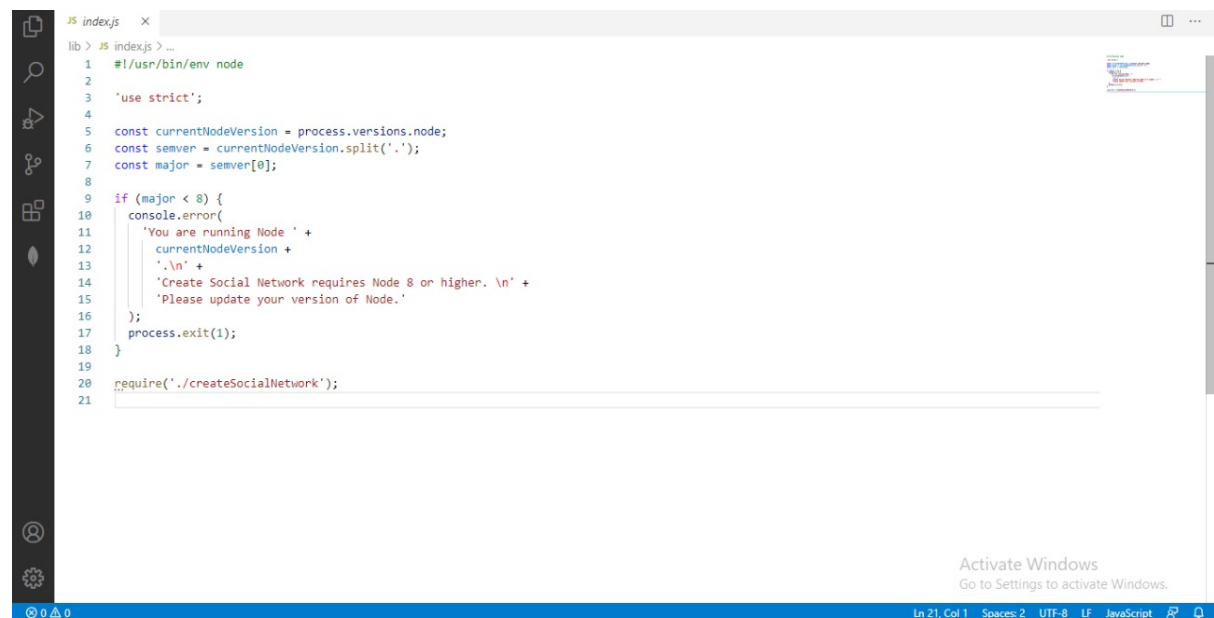
Index.js

```
const currentNodeVersion = process.versions.node;
const semver = currentNodeVersion.split('.');
const major = semver[0];

if (major < 8) {
  console.error(
    'You are running Node ' +
    currentNodeVersion +
    '.\n' +
    'Create Social Network requires Node 8 or higher. \n' +
    'Please update your version of Node.'
  );
  process.exit(1);
}

require('./createSocialNetwork');
```

Index.js



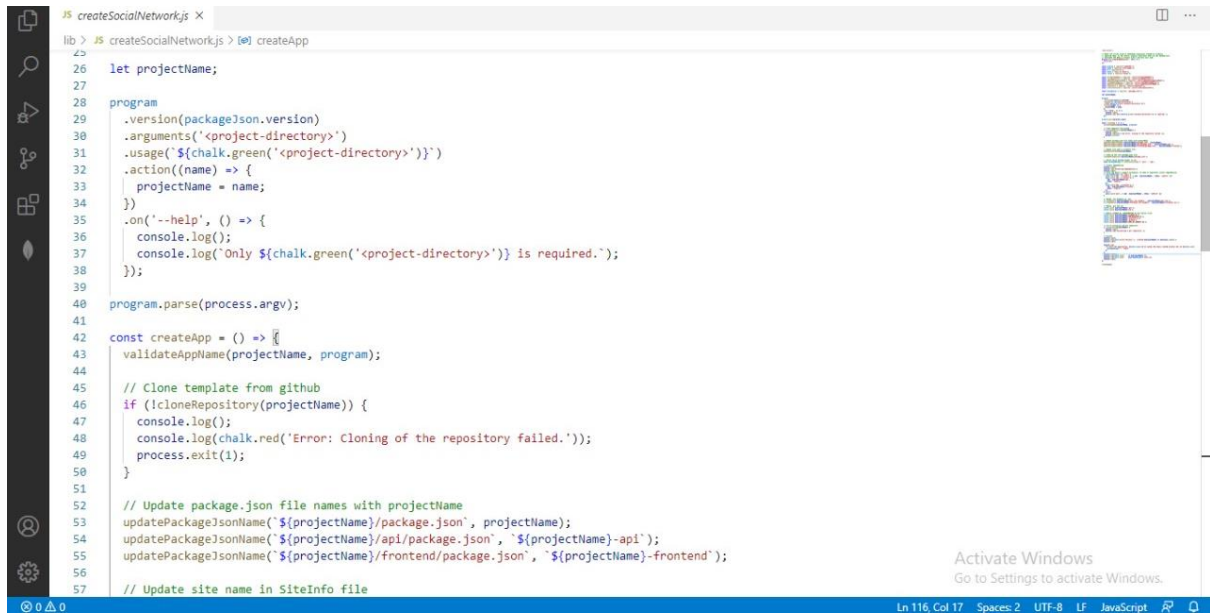
```
1  #!/usr/bin/env node
2
3  'use strict';
4
5  const currentNodeVersion = process.versions.node;
6  const semver = currentNodeVersion.split('.');
7  const major = semver[0];
8
9  if (major < 8) {
10   console.error(
11     'You are running Node ' +
12     currentNodeVersion +
13     '.\n' +
14     'Create Social Network requires Node 8 or higher. \n' +
15     'Please update your version of Node.'
16   );
17   process.exit(1);
18 }
19
20 require('./createSocialNetwork');
```

lib > JS index.js > ...

Activate Windows
Go to Settings to activate Windows.

Ln 21, Col 1 Spaces: 2 UTF-8 LF JavaScript

CreateSocialNetwork.js

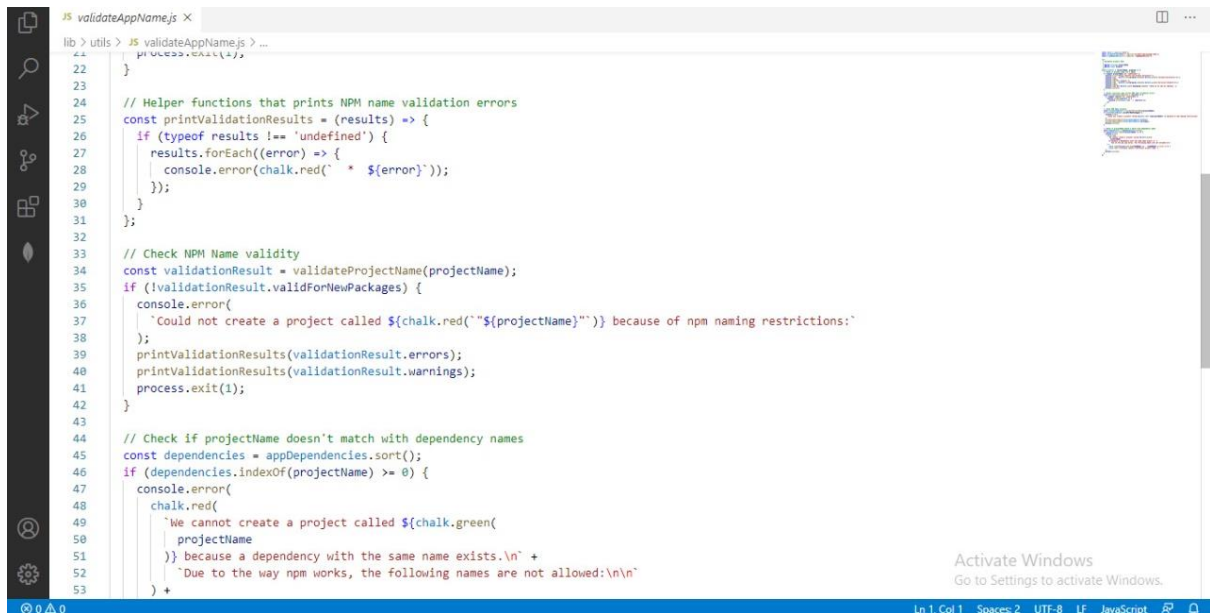


The screenshot shows a VS Code editor window with the file `createSocialNetwork.js` open. The code is a JavaScript script for creating a social network application. It includes a `program` object with `version`, `arguments`, `usage`, and `action` methods. The `action` method takes a `name` parameter and sets `projectName`. It also has a `--help` option. The `createApp` function calls `validateAppName` and then clones a repository from GitHub. It updates the `package.json` file with the project name and updates the site name in the `SiteInfo` file.

```
lib > JS createSocialNetwork.js > [0] createApp
25
26 let projectName;
27
28 program
29   .version(packageJson.version)
30   .arguments('<project-directory>')
31   .usage(`${chalk.green('<project-directory>')}`)
32   .action((name) => {
33     projectName = name;
34   })
35   .on('--help', () => {
36     console.log();
37     console.log(`Only ${chalk.green('<project-directory>')} is required.`);
38   });
39
40 program.parse(process.argv);
41
42 const createApp = () => {
43   validateAppName(projectName, program);
44
45   // Clone template from github
46   if (!cloneRepository(projectName)) {
47     console.log();
48     console.log(chalk.red('Error: Cloning of the repository failed.'));
49     process.exit(1);
50   }
51
52   // Update package.json file names with projectName
53   updatePackageName(`${projectName}/package.json`, projectName);
54   updatePackageName(`${projectName}/api/package.json`, `${projectName}-api`);
55   updatePackageName(`${projectName}/frontend/package.json`, `${projectName}-frontend`);
56
57   // Update site name in SiteInfo file
```

Ln 116, Col 17 · Spaces: 2 · UTF-8 · LF · JavaScript

Validate file



The screenshot shows a VS Code editor window with the file `validateAppName.js` open. The code is a JavaScript script for validating the project name. It includes a `printValidationResults` function that prints NPM name validation errors. It also includes a `validateProjectName` function that checks if the project name is valid for new packages. The `validateProjectName` function calls `validateProjectName` and prints the validation results. It also checks if the project name doesn't match with dependency names and prints an error message if it does.

```
lib > utils > JS validateAppName.js > ...
22
23
24 // Helper functions that prints NPM name validation errors
25 const printValidationResults = (results) => {
26   if (typeof results !== 'undefined') {
27     results.forEach((error) => {
28       console.error(chalk.red(` * ${error}`));
29     });
30   }
31 };
32
33 // Check NPM Name validity
34 const validationResult = validateProjectName(projectName);
35 if (!validationResult.validForNewPackages) {
36   console.error(
37     `Could not create a project called ${chalk.red(`${projectName}`)} because of npm naming restrictions:`
38   );
39   printValidationResults(validationResult.errors);
40   printValidationResults(validationResult.warnings);
41   process.exit(1);
42 }
43
44 // Check if projectName doesn't match with dependency names
45 const dependencies = appDependencies.sort();
46 if (dependencies.indexOf(projectName) >= 0) {
47   console.error(
48     chalk.red(
49       `We cannot create a project called ${chalk.green(
50         projectName
51       )} because a dependency with the same name exists.\n` +
52       `Due to the way npm works, the following names are not allowed:\n\n`
53     ) +
```

Ln 1, Col 1 · Spaces: 2 · UTF-8 · LF · JavaScript


```
lib > utils > JS tryGitinit.js > <unknown> > module.exports
1  const execSync = require('child_process').execSync;
2  const rimraf = require('rimraf');
3
4  /**
5   * Tries to initialize git repository in freshly created app
6   */
7  module.exports = (projectName) => {
8    let didInit = false;
9
10   try {
11     execSync('git --version', { stdio: 'ignore' });
12     execSync('git init', { cwd: projectName, stdio: 'ignore' });
13     didInit = true;
14
15     execSync('git add -A', { cwd: projectName, stdio: 'ignore' });
16     execSync('git commit -m "Initial commit from Create Social Network"', {
17       cwd: projectName,
18       stdio: 'ignore',
19     });
20   } catch (error) {
21     if (didInit) {
22       // If we successfully initialized but couldn't commit, maybe the commit author config is not set.
23       try {
24         rimraf.sync(`${projectName}/.git`);
25       } catch (removeErr) {
26         // Ignore.
27       }
28     }
29     return false;
30   }
31 };
32
```

Activate Windows
Go to Settings to activate Windows.

Ln 23, Col 12 Spaces: 2 UTF-8 LF JavaScript