Appendix C

main.js

```
const electron = require('electron');
const { app, BrowserWindow, ipcMain, Notification } = electron;

require('./database')

var mySql = require('mysql2');
const { getConnection } = require('./database');
var connection = mySql.createConnection({
  host: 'localhost',
  port: '3306',
  user: 'root',
  password: [redacted]
  database: 'taskmanager_electron'
});

connection.connect();

let mainWindow;
let addWindow;
let editWindow;

//function for creating the main window
function createMainWindow() {
  mainWindow = new BrowserWindow({
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    }
  });

  mainWindow.webContents.openDevTools();
  mainWindow.loadFile('index.html');
  mainWindow.on('closed', () => {
    mainWindow = null;
  });
}

app.on('ready', createMainWindow);

// Executed when command is sent from taskmanager.js when add button is clicked
```

```
ipcMain.on("createAddWindow", async () => {
  console.log("Received createAddWindow message");
  createAddWindow();
});

//Function to create Add window
function createAddWindow() {
  addWindow = new BrowserWindow({
    width: 800,
    height: 600,
    title: "Add New Task",
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
      menuIsEnabled: false,
    },
  });
  console.log("done");
  addWindow.loadFile("add.html");
  addWindow.on("closed", () => {
  addWindow = null;
  });
}

// Executed when command is sent from taskmanager.js when edit button is clicked
ipcMain.on("createEditWindow", (event, task) => {
  console.log("Received createEditWindow message", task);
  createEditWindow(task);
});


// Function to create Edit Window
function createEditWindow(task) {
  editWindow = new BrowserWindow({
    width: 800,
    height: 600,
    title: "Edit Task",
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    },
  });

  editWindow.loadFile("edit.html");
  editWindow.webContents.send("task:edit", task);
```

```
  editWindow.on("closed", () => {
    editWindow = null;
  });
}

ipcMain.on("display:DeletedTask", async(event) => {
  mainWindow.loadFile('TaskManager.html')
})

//Code to update task data in the info data base

ipcMain.on("update-form-data", async (event, formData) => {


  console.log('Received submit message with data:', formData);
  try {
    console.log('Received signin:submit message with data:', formData);
    // establish connection with mysql database
    const connection = await mySql.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });
    console.log('Connected to the database');
    //updating task data
    const result = await connection.execute(`UPDATE info SET Name = ?,duedate =
?, Priority = ?,TIME = ? WHERE Id = ?`,
    [formData.name, formData.date,formData.priority,formData.time,formData.id]);
    console.log('Query result:', result);
    connection.end();
    //closing the window once task is updated
    mainWindow.loadFile('TaskManager.html')
    editWindow.close();

  } catch (error) {
    console.error('Error in signin:submit event handler:', error);
  }
});

var mySql = require("mysql2");
var connection = mySql.createConnection({
  host: "localhost",
  port: "3306",
```

```
  user: "root",
  password: [redacted]
  database: "taskmanager_electron",
});

connection.connect();


const bcrypt = require('bcrypt');
//Function to add task info to the database
ipcMain.on("task:add", async (event, formData) => {
  console.log('Received add message with data:', formData);

  try {

    const connection = await mySql.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });
    console.log('Connected to the database');
    console.log(userId)
    //inserting data into the table
    const result = await connection.execute('INSERT INTO info (Name, duedate,
Priority, TIME, userid) VALUES (?,?,?,?,?)',
    [formData.name, formData.date,formData.priority,formData.time,userId]);

    console.log('Query result:', result);
    mainWindow.loadFile('taskManager.html')
    addWindow.close();
    //closing window when action has been performed

    connection.end();

  } catch (error) {
    console.error('Error in signin:submit event handler:', error);
  }
});


const mySqlPromise = require('mysql2/promise');

ipcMain.on("text:notes", async (event, text) => {
```

```
  console.log('Received add message with data:', text);

  try {
    const connection = await mySqlPromise.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });

    const [rows] = await connection.query("SELECT * FROM notes WHERE userid=?",
[userId]);
    if (rows.length > 0) {
      newText= text

      await connection.execute(`UPDATE notes SET notes = ? WHERE userid = ?`,
[newText, userId]);
      event.reply('text:display', newText)
      console.log(text)
    } else {

      newtext= text
      await connection.execute('INSERT INTO notes (userid, notes) VALUES (?,?)',
[userId, newText]);
      event.reply('text:display', newText)
    }

    notification = new Notification({
      title: "Note saved",
      body: "Your note has been saved in the database"}).show()

    await connection.end(); // Close the connection after use

  } catch (error) {
    console.error('Error:', error);
  }
});


ipcMain.on("open", async (event) => {

    const connection = await mySqlPromise.createConnection({
      host: 'localhost',
```

```
      port: '3306',
      user: 'root',
      password:[redacted]
      database: 'taskmanager_electron'
    });
    const[result] = await connection.query(`SELECT * FROM notes where userid=?`,
[userId])
    console.log(userId)
    console.log(result)
    const note = result[0].notes
    console.log(note)
    event.reply('text:display', note); // Send the message to the renderer
process
    console.log(note);
    });

const { dialog } = require('electron');
class User{
  constructor(email,password, name,table){
    this.email = email;
    this.password = password;
    this.name = name;
    this.table = table;
  }

  async getConnection() {
    try {
      const connection = await mySql.createConnection({
        host: 'localhost',
        port: '3306',
        user: 'root',
        password:[redacted]
        database: 'taskmanager_electron'
      });
      console.log('Connected to the database');
      return connection;
    } catch (error) {
      console.error('Error in getConnection:', error);
    }
  }


async signUp(data){
  try {
```

```
    const connection = await mySql.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });
    console.log('Connected to the database');
    const salt = bcrypt.genSaltSync(10);
    const hash = bcrypt.hashSync(data.password, salt);
    const result = await connection.execute(`INSERT INTO ${this.table} (email,
password, name) VALUES (?, ?, ?)`, [this.email, hash, this.name]);
    console.log('Query result:', result);
    connection.end(); // Close the connection after use

  } catch (error) {
    console.error('Error in signin:submit event handler:', error);
  }
}

async logIn(mainWindow, data, event) {
  try {
    const connection = await mySql.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });

    const[rows] = await connection.query(`SELECT * FROM ${this.table} where
email=?`, [this.email], function (error, rows, fields) {
      if (error) throw error;
      console.log(rows, 'rows', this.email, this.password)

      if (rows.length > 0) {
        const hash = rows[0].password;
        userId = rows[0].userid;
        email = rows[0].email;

      const plainPassword = data.password;

      const result = bcrypt.compareSync(plainPassword, hash);
      console.log(result);
      console.log(plainPassword)
```

```
      console.log(hash)

      if (result == true) {
        mainWindow.loadFile('home.html')}

      else {
        dialog.showErrorBox('Log in error', 'Incorrect Password');
      }

    }
    else{
      dialog.showErrorBox('Log in error', 'Incorrect Email');
    }
    connection.end();
  })

  }catch (error) {
    console.error('Error in login event handler:', error);
  }
}
}

let userId;

class Student extends User{
  constructor(email, password, name) {
    super(email, password, name, 'users');
  }}

let email;
class Admin extends User{
  constructor(email, password, name) {
    super(email, password, name,'admins');
  }
// establish connection with database
  async getConnection() {
    try {
      const connection = await mySql.createConnection({
        host: 'localhost',
        port: '3306',
        user: 'root',
        password: [redacted]
        database: 'taskmanager_electron'
      });
      console.log('Connected to the database');
```

```
      return connection;
    } catch (error) {
      console.error('Error in getConnection:', error);
    }
  }
//function for signing up
  async signUp(data){
    try {
      const connection = await mySql.createConnection({
        host: 'localhost',
        port: '3306',
        user: 'root',
        password: [redacted]
        database: 'taskmanager_electron'
      });
      //generating salt for hashing
      const salt = bcrypt.genSaltSync(10);
      const hash = bcrypt.hashSync(data.password, salt);
      //inserting hashed password into database
      const result = await connection.execute(`INSERT INTO admins (email,
password, name) VALUES (?, ?, ?)`, [this.email, hash, this.name]);
      console.log('Query result:', result);
      connection.end(); // Close the connection after use
    } catch (error) {
      console.error('Error in signin:submit event handler:', error);
    }
  }

  async logIn(mainWindow, data, event) {
    try {
      const connection = await mySql.createConnection({
        host: 'localhost',
        port: '3306',
        user: 'root',
        password: [redacted]
        database: 'taskmanager_electron'
      });

      const[rows] = await connection.query(`SELECT * FROM ${this.table} where
email=?`, [this.email], function (error, rows, fields) {
        if (error) throw error;
        console.log(rows, 'rows', this.email, this.password)
        if (rows.length > 0) {
          const hash = rows[0].password;
          const plainPassword = data.password;
```

```
          const result = bcrypt.compareSync(plainPassword, hash);
          console.log(result);
          console.log(plainPassword)
          console.log(hash)

        if (result == true) {
          mainWindow.loadFile('control.html')
        } else {
          dialog.showErrorBox('Log in error', 'Incorrect Password');
        }
      }
      else{
        dialog.showErrorBox('Log in error', 'Incorrect Email');
      }
      });
      connection.end();
    } catch (error) {
      console.error('Error in login event handler:', error);
    }

  }
}

ipcMain.on('signin:submit', async (event, data) => {
  console.log('Received submit message with data:', data);
  try {
    const student = new Student(data.email, data.password,data.name)
    await student.signUp(data);
  }
  catch(error){
    console.error('Error in signin:submit event handler:' ,error);
  }
});

ipcMain.on('login:submit', async (event, data) => {
  console.log("login received");
  try{
    const student = new Student(data.email, data.password, data.name);
    await student.logIn(mainWindow, data, event);
  }catch (error){
    console.error('error in login event handler', error);
  }
});

  ipcMain.on('new:admin', async(event, data) => {
```

```
    console.log("login receieved")
    try{
      const admin = new Admin(data.email, data.password, data.name);
      console.log('got data', data)
      await admin.signUp(data);
      console.log("admin sign in function initiated")
    } catch (error){
      console.error('new:Admin', event)
    }
  });

  ipcMain.on('admin:submit', async (event, data) => {
    console.log("login receieved")
    try {
        const admin = new Admin(data.email, data.password, data.name)
        await admin.logIn(mainWindow, data, event);
    }catch (error){
        console.error('Error admin:submit', error)
    }
  });

ipcMain.on('getUserId', (event) => {
    event.reply('userId', userId);
  });

ipcMain.on('getEmail',(event) => {
    event.reply('email', email)
    console.log(email)
  });

  ipcMain.on('getStudEmail',async(event, data) =>{
    console.log('sent')
    const connection = await mySqlPromise.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
    });


      const[rows] = await connection.query(`SELECT * FROM users where userid=?`,
[data])
      if (rows.length > 0) {
        thisEmail = rows[0].email
```

```
        event.reply("email", thisEmail)
        console.log(thisEmail)

      }})

ipcMain.on('update:key',(async (event, cKey) => {

  try{
    console.log('Received signin:submit message with data:', cKey);
    const connection = await mySql.createConnection({
      host: 'localhost',
      port: '3306',
      user: 'root',
      password: [redacted]
      database: 'taskmanager_electron'
      });
    console.log('Connected to the database');
    const result = await connection.execute(`UPDATE admins SET \`key\`
=?`,[cKey]);
    console.log('Query result:', result);
    connection.end(); // Close the connection after use
    event.reply('key:updated')

  } catch (error) {
    console.error('Error in signin:submit event handler:', error);
  }
})
),

ipcMain.on('get:key',async(event, data) => {
  console.log("got key message")
  const connection = await mySql.createConnection({
    host: 'localhost',
    port: '3306',
    user: 'root',
    password: [redacted]
    database: 'taskmanager_electron'
  });

    console.log(data)

    const[rows] = await connection.query(`SELECT * FROM admins where email=?`,
[data], function (error, rows, fields) {
    if (error) throw error;
      console.log('rows are:', rows)
```

```
    if (rows.length > 0) {
      key = rows[0].key
      event.reply('key', key)
      console.log("key is", key)
      }
    })
  })

ipcMain.on("Notify",(event)=>{
  console.log("received notif")
  notification = new Notification({
  title: "Email sent",
  body: "Email has been sent"}).show()
})

let studentId
ipcMain.on("openCal", async(event, data)=>{
  mainWindow.loadFile('adminCalendar.html')
  studentId = data
  })

ipcMain.on('getStudentId', async(event)=>{
  event.reply("sentStudentId", studentId)
})

ipcMain.on('reloadDeleteAdmin', async(event)=>{
  mainWindow.loadFile("control.html")
})

module.exports = {
  createMainWindow,
};
```

index.html

```
<header>
    <link rel="stylesheet" href="index.css"/>

    <script>require('./home')</script>

</header>

<body>
    <h1>Choose login</h1>
```

```
    <div>
        <h2>Welcome</h2>
    </div>

    <div class="home">
        <a href="login.html">
            <button class="button" type="submit">Student login</button>
        </a>
    </div>

    <div class="home">
        <a href="admin.html">
            <button class="button1" type="submit">Admin login</button>
        </a>
    </div>
</body>
```

index.css

```
.message{
    position: absolute;
    bottom: 70;
    right: 340;
}

.button {

    font-size: 20;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(244, 244, 244);
    padding: 20px 30px;
    border-radius: 20px;
    background-color:#921bed;
    margin-bottom: 50px;
}

.button1 {

    font-size: 20;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(252, 247, 247);
    padding: 20px 30px;
    border-radius: 20px;
    background-color:#921bed;
```

```
    margin-bottom: 300px
}

h1 {
    text-align: center;
    margin-top: 50px;
    margin-bottom: 50px;
}

h2 {
    text-align: center;
    margin-bottom: 50px;
}

.home {
    text-align: center;
     margin-bottom: 20px;
}

.back{
    position: absolute;
    bottom: 60;
    right: 360;
}
```

login.html:

```
<<head>

    <link rel="stylesheet" href="login.css"/>
    <script src= login.js> </script>

  <body>
    <h1> Log in Page</h1>
    <form class="loginForm" id="loginForm">
      <div class="form">
        <label for="Email">Enter email: </label>
        <input type="email" name="Email" id="email" autofocus="" required>
      </div>
      <div class="form">
        <label for="Password">Enter Password: </label>
        <input type="password" name="Password" id="password"
autofocus=""  title="Must contain at least one number and one uppercase and
lowercase letter, and at least 8 or more characters" required>
```

```
      </div>

        <div>
            <button class="button" type="submit">
           Log in
       </button>

          </form>
  <div id="message"> </div>


  <h3 class="message">New user?</h3>

               <div class="signup">
                 <a href="signIn.html">Sign up</a>

                 <div class="signup">
                   <a href="index.html">Back</a>
              </div>

        </div>
```

login.css:

```
input {

    background-color:#C4E4FF ;
    margin-top: 30px;
    margin-bottom: 30px
}


label {
    padding-right: 20px;
    font-size: 20px;
}

h1 {
    text-align: center;
    margin-top: 100px;
    margin-bottom: 20px;
}
```

```css
.form {
    text-align: center;
}

.loginform {
    text-align: center;
}

.button {
    text-align: center;
    color: rgb(249, 238, 238);
    font-family: 'Times New Roman', Times, serif;
    font-size: 20px;
    border-radius: 20px;
    background-color:blueviolet;
    padding: 10px 20px;
    margin-bottom: 40px;
}


 .signup{
    margin-top: 25px
 }

 }
```

login.js:

```js
const { ipcRenderer } = require('electron');
const { BrowserWindow } = require('electron');

document.addEventListener('DOMContentLoaded', () => {
    const loginForm = document.querySelector('.loginForm')
    console.log(loginForm)
    loginForm.addEventListener('submit', async (event) => {
    event.preventDefault();
    console.log('submitted')
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    ipcRenderer.send('login:submit', { email, password });
    console.log('sent')
});
```

```
ipcRenderer.on('login:error', (event, errorMessage) => {
    // Display an error message to the user or handle it in your UI
    console.log(errorMessage);
    message.textContent = 'Email or password is incorrect!'
})

})
```

signIn.html:

```
<head>
    <link rel="stylesheet" href="signIn.css"/>
</head>

<body>
    <h1> Sign in Page</h1>
    <form method="get" class="form">
        <div class="form">
          <label for="Email">Enter email: </label>
          <input type="email" name="Email" id="email" required>
        </div>

        <div class="form">
          <label for="password">Enter Password: </label>
          <input type="text" name="password" id="password"
pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
          title="Must contain at least one number and one uppercase and lowercase
letter, and at least 8 or more characters" required>
        </div>

        <div>
          <label for="confirmPassword">Confirm Password:</label>
          <input type="password" id="confirmPassword" name="confirmPassword"
required>
        </div>

        <div>
          <label for="name">Name:</label>
          <input  id="name" name="name" required>
        </div>

        <div id="message"></div>
        <div id="success"></div>

        <button class="button" type="submit" id="signupForm">
```

```
          Sign up
        </button>

        <div class="back">
                <a href="login.html">Login</a>
         </div>
        </div>
    </form>

    <script>require('./signIn.js')</script>
    </body>
```

signIn.css:

```
input {
    background-color:#C4E4FF;
    margin-bottom: 50px;
}


label {
    padding-right: 10px;
    font-size: 20px;
}

h1 {
    text-align: center;
    margin-top: 75px;
    margin-bottom: 30px;
}

.form {
    text-align: center;
}

.button {
    text-align: center;
    color: rgb(249, 238, 238);
    font-family: 'Times New Roman', Times, serif;
    font-size: 20px;
    border-radius: 20px;
    background-color: #921bed;
    padding: 10px 20px;
    margin-bottom: 40px;
}
```

signIn.js:

```
const electron = require('electron');
const { ipcRenderer } = electron;

document.querySelector('.form').addEventListener('submit', async (event) => {
    event.preventDefault();
    console.log('Submit button clicked');
    // Fetch form values
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const confirmPassword = document.getElementById('confirmPassword').value;
    const name = document.getElementById('name').value

        // Send data to main process
        if (password === confirmPassword) {
        console.log('here')
        ipcRenderer.send('signin:submit', { email, password, name });
            message.textContent = ''
            success.textContent = 'Submission successful!';
            document.getElementById('email').value = '';
            document.getElementById('password').value = '';
            document.getElementById('confirmPassword').value = '';
            document.getElementById('name').value = ''

        } else {
            message.textContent = 'Passwords do not match!';
        }
});
```

admin.html:

```
<head>
    <link rel="stylesheet" href="login.css"/>

  <body>
    <h1> Admin login</h1>
    <form class="loginform">
      <div class="form">
        <label for="Email">Enter email: </label>
        <input type="email" name="Email" id="email" autofocus="" required>
      </div>
```

```
      <div class="form">
        <label for="Password">Enter Password: </label>
        <input type="password" name="Password" id="password"
autofocus=""  title="Must contain at least one number and one uppercase and
lowercase letter, and at least 8 or more characters" required>
      </div>


        <div>
            <button class="button"
            type="submit" id="loginform">
           Log in
       </button>

          </form>
  <div id="message"> </div>

  <div class="signup">
    <a href="adminSignIn.html">Create new admin</a>
</div>
                <div class="signup">
                  <a href="index.html">Back</a>
              </div>



        </div>

        <script>require('./admin.js')</script>
```

admin.js:

```
const { ipcRenderer } = require('electron');

document.querySelector('.loginform').addEventListener('submit', async (event) =>
{
    event.preventDefault();
    console.log('submitted')
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    ipcRenderer.send('admin:submit', { email, password });
    console.log('sent')
});
```

```
ipcRenderer.on('admin:error', (event, errorMessage) => {
    console.log(errorMessage);
    message.textContent = 'Access not granted!'
})
```

adminSignIn.html:

```
<head>
    <link rel="stylesheet" href="signIn.css"/>
</head>

<body>
    <h1> Create new admin</h1>
    <form method="get" class="form">
        <div class="form">
          <label for="Email">Enter email: </label>
          <input type="email" name="Email" id="email" required>
        </div>
        <div class="form">
          <label for="Password">Enter Password: </label>
          <input type="text" name="password" id="password"
pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Must contain at least one
number and one uppercase and lowercase letter, and at least 8 or more characters"
required>
        </div>
        <div>
        <label for="confirmPassword">Confirm Password:</label>
        <input type="password" id="cp" name="confirmPassword" required>
        <div>
          <div>
        <label for="name">Name:</label>
        <input id="name" name="name" required>
        <div>
        <label for="adminKey">Admin key:</label>
        <input type="text" id="key" name="adminKey" required>
        </div>

        <div id="message"></div>
        <div id="success"></div>

            <button class="button"
                 type="submit" id="signupForm">
```

```
                Create
            </button>

        <div class="back">
                <a href="admin.html">Login</a>

         </div>


        </div>
    </form>

    <script>require('./adminSignIn.js')</script>
    </body>
```

adminSignIn.js:

```
const electron = require('electron');
const { ipcRenderer } = electron;


document.querySelector('.form').addEventListener('submit', async (event) => {
    event.preventDefault();
    console.log('Submit button clicked');
    // Fetch form values
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const confirmPassword = document.getElementById('confirmPassword').value;
    const adminKey = document.getElementById('key').value;
    const name = document.getElementById('name').value;
        // Send data to main process
    if (password === confirmPassword) {
        console.log('passwords match')
        data = 'newadmin@23'
        ipcRenderer.send('get:key', data)
        ipcRenderer.on('key', (event, key) => {
        confirmKey = key
        console.log(confirmKey)

        if (adminKey === confirmKey){
                console.log(adminKey)
                ipcRenderer.send('new:admin', { email, password, name });
                message.textContent = ''
                success.textContent = 'Submission successful!';
                document.getElementById('email').value = '';
```

```
                document.getElementById('password').value = '';
                document.getElementById('confirmPassword').value = '';
                document.getElementById('key').value = '';
                document.getElementById('name').value = '';
        }

        else{
            console.log("admin key is"+ adminKey)
            message.textContent = 'Wrong admin key'
            }

        })

    } else {
        console.log('dont match')
        message.textContent = 'Passwords do not match!';
    }
});
```

home.html:

```
<header>
    <link rel="stylesheet" href="home.css"/>

    <script>require('./home')</script>

</header>

<body>
    <h1>Home Page</h1>

    <div>
        <h2>Welcome</h2>
    </div>

    <div class="home">
        <a href="taskManager.html">
            <button class="button" type="submit">Task Manager</button>
        </a>
    </div>

    <div class="home">
        <a href="calendar.html">
```

```
            <button class="button1" type="submit">Calendar</button>

        </a>
    </div>

    <div class="back">
        <a href="login.html">Signout</a>

    </div>

</body>

<script>

const ipcRenderer = require('electron').ipcRenderer;

</script>
```

home.css:

```
home {
    text-align: center;
     margin-bottom: 20px;
}

.button {

    font-size: 20;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(255, 251, 251);
    padding: 20px 30px;
    border-radius: 20px;
    background-color: #921bed;
    margin-bottom: 50px;
}

.button1 {
    font-size: 20;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(246, 240, 240);
    padding: 20px 30px;
    border-radius: 20px;
    background-color:#921bed;
    margin-bottom: 300px
}
```

```css
h1 {
    text-align: center;
    margin-top: 50px;
    margin-bottom: 50px;
}

h2 {
    text-align: center;
    margin-bottom: 50px;
}
```

taskManager.html:

```html
<head>
    <link rel="stylesheet" href="taskManager.css" />
  </head>

  <body>
    <h1>Task Manager</h1>

    <div id="task">
      <table id="myTable">
        <thead>

          <th> Name </th>
          <th> Date </th>
          <th> Priority </th>
          <th> Time </th>


        </thead>
        <tbody id="taskList">

        </tbody>
      </table>

    </div>

    <button class="button" type="submit" id="createAddWindow" style="cursor:
     pointer;" onlick="createAddWindow()">Add new
      <div>+</div>

    </button>
```

```
    <div class="back">
      <a href="home.html">Back</a>
    </div>

    <script>require('./taskManager.js')</script>

  </body>
```

taskManager.css:

```
.formbox {
    border: 1px solid black;
    background-color: rgba(164, 89, 235, 0.534);
    position: absolute;
    bottom: 0;
    right: 50;

  }

  .button {
    font-size: 25px;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(240, 233, 233);
    padding: 20px 20px;

    background-color: rgb(171, 80, 252);
    margin-top: 50px;
    margin-left: 300px;

  }

  .back{
    margin-top: 50px;
  }

 h1 {
    text-align: center;
    margin-top: 50px;
    margin-bottom: 50px;
}

h2 {
    text-align: center;
}
```

```
#myTable {
font-family: 'Times New Roman', Times, serif;
border-collapse: collapse;
width: 100%;
margin-top: 50px;
margin-bottom: 20px;
}

#myTable td, #myTable th {
border: 1px solid #ddd;
padding: 8px;
}

#myTable tr:nth-child(even){background-color: #f2f2f2;}

#myTable tr:hover {background-color: #ddd;}

#myTable th {
padding-top: 12px;
padding-bottom: 12px;
text-align: left;
background-color: #45ffd1;
color: rgb(0, 0, 0);
margin-top: 20px;

}



input {
background-color:#C4E4FF;
font-family: 'Times New Roman', Times, serif;
margin: 10px;
padding-left: 20px;
}



label {
padding-left: 5px;
font-size: 20px;
}

#tl li {
```

```
margin-bottom: 20px;
}

#tasks{
margin-bottom: 20px;
}

.delBtn {
font-size: 15;
font-family: 'Times New Roman', Times, serif;
color: rgb(255, 255, 255);
padding: 6px 8px;
margin-right: 10px;
margin-left: 10px;
border-radius: 5px;
background-color: rgb(171, 80, 252)
}

.edtBtn{
font-size: 15;
font-family: 'Times New Roman', Times, serif;
color: rgb(255, 255, 255);
padding: 6px 8px;
margin-right: 10px;
margin-left: 10px;
border-radius: 5px;
background-color: rgb(171, 80, 252)
}

.calBtn{
    font-size: 15;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(255, 255, 255);
    padding: 6px 8px;
    margin-right: 10px;
    margin-left: 10px;
    border-radius: 5px;
    background-color: rgb(171, 80, 252)
    }

.set{
    font-size: 15;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(255, 255, 255);
    padding: 6px 8px;
```

```
    margin-right: 10px;
    margin-left: 10px;
    border-radius: 5px;
    background-color: rgb(171, 80, 252)
    }
```

taskManager.js:

```
const ipcRenderer = require('electron').ipcRenderer;
const moment = require('moment');

ipcRenderer.send('getUserId');
console.log("Sent")

ipcRenderer.on('userId', (event, userId) =>{
  console.log("Receieved")
  uid = userId
  console.log('Received user ID:', userId);


const { getConnection } = require('./database');
const con = getConnection()
console.log(con)

con.connect(function (err) {
  if (err) throw err;
  con.query("SELECT * FROM taskmanager_electron.info WHERE userid = ?", [uid],
function (err, result, fields) {
    if (err) throw err;
      if (result.length > 0) {
        renderTask(result);
        console.log(result)
    }
  });
});


class Node{
  constructor(task){
    this.task = task;
    this.next = null;
  }
}
class LinkedList {
  constructor() {
```

```
    this.head = null;
    this.tail = null;
  }
  insert(task) {
    const newNode = new Node(task);

    // If the list is empty, simply add the new node as both head and tail
    if (!this.head) {
      this.head = newNode;
      this.tail = newNode;
      return;
    }
    let current = this.head;
    let prev = null;
    // Traverse the list to find the correct position based on due date
    while (current && new Date(current.task.duedate) < new Date(task.duedate)) {
      prev = current;
      current = current.next;
    }
    // Adjust task priorities to numeric values
    switch (task.Priority) {
      case "High":
        task.Priority = 3;
        break;
      case "Medium":
        task.Priority = 2;
        break;
      case "Low":
        task.Priority = 1;
        break;
      default:
        task.Priority = 0; // Default to lowest priority
        break;
    }
    // If there are tasks due on the same date, sort them by priority
    while (current && new Date(current.task.duedate).getTime() === new
Date(task.duedate).getTime()) {
      if (current.task.Priority < task.Priority) {
        break;
      }
      prev = current;
      current = current.next;
    }

    // Insert the new node at the correct position
```

```
    if (!prev) {
      newNode.next = this.head;
      this.head = newNode;
    } else {
      prev.next = newNode;
      newNode.next = current;
      if (!current) {
        this.tail = newNode;
      }
    }
  }


  // Other methods...
  convertPrioritiesToString() {
    let current = this.head;
    while (current) {
      this.convertPriorityToString(current.task);
      current = current.next;
    }
  }

  convertPriorityToString(task) {
    switch (task.Priority) {
      case 3:
        task.Priority = "High";
        break;
      case 2:
        task.Priority = "Medium";
        break;
      case 1:
        task.Priority = "Low";
        break;
    }
  }
}


function renderTask(result) {
  console.log('Received task data:', result);
  const linkedList = new LinkedList();
  result.forEach(task => {
  linkedList.insert(task)
})
```

```
  let current = linkedList.head; // Start traversal from the head of the linked
list
  console.log(current)
  while (current) {
    console.log(current)
    const task = current.task;
    linkedList.convertPrioritiesToString();
    taskList.innerHTML += `
      <tr>
        <td>${task.Name}</td>
        <td>${moment(task.duedate).format("MMM-Do-YYYY")}</td>
        <td>${task.Priority}</td>
        <td>${task.TIME}</td>
        <td><button id="delBtn" class ="delBtn" data-id="${task.Id}"
onclick="deleteTask(${task.Id})">Delete</button></td>
        <td><button id="edtBtn" class ="edtBtn" data-id="${task.Id}"
onclick="editTask(${task.Id})">Edit</button></td>
      </tr>
          `
      current = current.next
  }

  console.log(task.Priority)
}

document.addEventListener("DOMContentLoaded", function () {
  const taskList = document.getElementById('taskList');

  if (taskList) {
    taskList.addEventListener('click', (event) => {
      const target = event.target;

      if (target.tagName === 'BUTTON' && target.classList.contains('delBtn')) {
        const data = target.getAttribute('data-Id');
        deleteTask(data);
      }
      // Add similar logic for the edit button if needed
      if (target.tagName === 'BUTTON' && target.classList.contains('edtBtn')) {
        const data = target.getAttribute('data-Id');
        editTask(data);
      }
    });
  }
});
```

```
function deleteTask(data) {
  con.connect(function (err) {
    if (err) throw err;
    const response = confirm(
      "Are you sure you want to delete Task ?"
    );
    if (response) {
      con.query(
        "DELETE FROM taskmanager_electron.info where Id=?", [data],
        function (err, result, fields) {
          if (err) throw err;
        }
      );
    }
    ipcRenderer.send("display:DeletedTask");
  });
}

function editTask(data) {
  console.log(data, 'data')
  con.connect(function (err) {
    if (err) throw err;
    con.query(
      "SELECT * FROM taskmanager_electron.info where Id=?",[data],
      function (err, result, fields) {
        if (err) throw err;
        ipcRenderer.send("createEditWindow", result);
      }
    );
  });
}

const deleteBtn = document.querySelector('.delBtn');
if (deleteBtn) {
  deleteBtn.addEventListener('click', () => {
    deleteTask(task.Id);
  });
}

const editBtnId = document.querySelector('.edtBtn');
if (editBtnId) {
  editBtnId.addEventListener('click', () => {
    editTask(task.Id);
  });
}
```

```
})
const createAddWindow = document.querySelector('.button');
createAddWindow.addEventListener('click', (event) => {
  console.log(event, 'Add Button')
  ipcRenderer.send('createAddWindow');
});
```

add.html:

```
<head>
    <link rel="stylesheet" href="add.css" />


</head>

<body>
    <h1>
        Add new task
    </h1>
    <form id="addform" class="addform">
        <div>
            <label for="name">Name:</label>
            <input id="name" value="" name="name" required>
        </div>

        <div>
            <label for="duedate">Due Date:</label>
            <input type="date" id="duedate" value="" name="duedate" required>
        </div>

        <div>
            <label for="priority">Priority of task:</label>
            <select id="priority" name="priority" required>

                <option> High </option>
                <option> Medium </option>
                <option> Low </option>

            </select>
        </div>

        <div>
            <label for="time">Time to complete (HH:MM:SS):</label>
            <input id="time" value="" name="time">
```

```
        </div>
        <button class="but" type="submit" id="createTask">Add Task</button>

    </form>

    <script>require('./add.js')</script>

</body>
```

add.css:

```
label{
    padding-right: 5px;
    font-size: 20px;
    padding-left: 5px;
}

input, select {
    background-color:#C4E4FF;
    margin-bottom: 30px;
    padding-left: 20px;
    font-family: 'Times New Roman', Times, serif;
}



.but{
    font-size: 15px;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(255, 254, 254);
    padding: 10px 20px;
    border-radius: 20px;
    background-color: rgb(185, 49, 252);
    position: absolute;
    top: 300;
    left: 175;


}

h1 {
    text-align: center;
}
```

add.js

```
const { ipcRenderer } = require('electron');

ipcRenderer.send('getUserId');

ipcRenderer.on('userId', (event, userId) => {
  uid = userId
  console.log('Received user ID:', userId);
});

document.querySelector('.addform').addEventListener('submit', async (event) => {
  event.preventDefault(); // Prevent the default form submission behavior

  // Collect form data
  const formData = {
      name: document.getElementById('name').value,
      date: document.getElementById("duedate").value,
      priority: document.getElementById("priority").value,
      time: document.getElementById("time").value,
      userid: uid,
  }

  // Send the form data to the main process
  ipcRenderer.send('task:add', formData);
  console.log('sent add task')
})
```

edit.html:

```
<head>
  <link rel="stylesheet" href="edit.css" />
</head>

<body>
  <h1>Edit Task</h1>

  <form id="editForm" class="editForm">
    <div>
      <label for="name">Name:</label>
      <input id="name" value="" name="name">
```

```
    </div>

    <div>
      <label for="duedate">Due Date:</label>
      <input type="date" id="duedate" value="" name="duedate">
    </div>

    <div>
      <label for="priority">Priority of task:</label>
      <select id="priority" name="priority">

        <option> High </option>
        <option> Medium </option>
        <option> Low </option>

      </select>
    </div>

    <div>
      <label for="time">Time to complete:</label>
      <input id="time" value="" name="time">
    </div>

    <button class="but" type="submit">Save</button>
  </form>
  <script>require('./edit.js')</script>
</body>
```

edit.css

```
label{
    padding-right: 5px;

    font-size: 20px;
    padding-left: 5px;

}

input, select {
    background-color:rgb(255, 213, 185);

    margin-bottom: 30px;
    padding-left: 20px;
}
```

```
.but{
    font-size: 15px;
    font-family: 'Times New Roman', Times, serif;
    color: rgb(8, 8, 8);
    padding: 10px 20px;
    border-radius: 20px;
    background-image: linear-gradient(to right, rgb(88, 48, 221), rgb(178, 34,
194) 50%, rgb(230, 5, 110));
    position: absolute;
    top: 300;
    left: 150;
}

h1 {
    text-align: center;
}
```

edit.js

```
const { ipcRenderer } = require('electron');
const moment = require('moment');
var editId;


// Listen for the 'populate-form' event from the main process
ipcRenderer.on('task:edit', (event, task) => {
    // Populate form fields with the received data
    document.getElementById('name').value = task[0].Name || '';
    document.getElementById("time").value = task[0].TIME || ''
    let dueDate = moment(task[0].duedate).format('YYYY-MM-DD')
    document.getElementById("duedate").value = dueDate || '';
    document.getElementById('priority').value = task[0].Priority || '';
    editId = task[0].Id
});

document.querySelector('.editForm').addEventListener('submit', async (event) => {
    event.preventDefault(); // Prevent the default form submission behavior

    // Collect form data
    const formData = {
        name: document.getElementById('name').value,
        date: document.getElementById("duedate").value,
        priority: document.getElementById("priority").value,
        time: document.getElementById("time").value,
```

```
        id: editId
    };

    // Send the form data to the main process
    ipcRenderer.send('update-form-data', formData);
    console.log('sent edit task')
})
```

calendar.html

```
<head>
    <link rel="stylesheet" href="calendar.css"/>
    <script
src='https://cdn.jsdelivr.net/npm/fullcalendar@6.1.8/index.global.min.js'></scrip
t>
    <script>src="https://cdn.jsdelivr.net/npm/quill@2.0.0-
rc.4/dist/quill.js"</script>
    <link rel='stylesheet'
href='https://fullcalendar.io/releases/core/4.0.2/main.min.css'>
     <link rel='stylesheet'
href='https://fullcalendar.io/releases/daygrid/4.0.1/main.min.css'>
     <link href="https://cdn.jsdelivr.net/npm/quill@2.0.0-
rc.4/dist/quill.snow.css" rel="stylesheet" />


</head>
    <body>

        <h1> Calendar </h1>

        <div id='calendar'></div>
        <div id="editor"></div>

<script  src="./calendar.js"></script>
<script src='https://fullcalendar.io/releases/core/4.0.2/main.min.js'></script>
  <script
src='https://fullcalendar.io/releases/daygrid/4.0.1/main.min.js'></script>
  <script src='https://fullcalendar.io/releases/list/4.0.1/main.min.js'></script>
  <script src='https://fullcalendar.io/releases/google-
calendar/4.0.1/main.min.js'></script>


        <div class="back">
            <a href="home.html">Back</a>
```

```
        </div>
        </div>

    </body>
```

calendar.css

```
h1{
  text-align: center;
}

.back{
  margin-top: 60px;
}


.text{
  padding: 60px 50px;
  background-color: rgb(253, 251, 255);
  margin: 10px;
  color:#141414;
  word-wrap: break-word;
  font-family: 'Times New Roman', Times, serif;
  font-size: 18px;

}

#editor {
  width: 600px;
  height: 300px;
  margin-left:10px;
}


.ql-toolbar {
  width: 600px;
  margin-top:100px;
  margin-left:10px;
  background-color: #96E9C6;;
}

.ql-save-button{
  margin-right: 10px;
}
```

```
#calendar .fc-toolbar.fc-header-toolbar button {
  background-color: #921bed;
  color: #fffafa;
}
```

calendar.js

```
const{getConnection} = require('./database');
const con =  getConnection()


const ipcRenderer = require('electron').ipcRenderer;

ipcRenderer.send('open')

ipcRenderer.send('getUserId');




ipcRenderer.on('userId', (event, userId) => {
uid = userId
console.log('Received user ID:', uid);



const customEvents = []
con.connect(function (err) {
  if (err) throw err;
  con.query("SELECT * FROM taskmanager_electron.info WHERE userid = ?", [uid],
function (err, result, fields) {
    if (err) throw err;
    if (result.length > 0) {
      result.forEach((res) => {
        customEvents.push({
          title: res.Name,
          start: new Date(res.duedate).toISOString(),
          end: new Date(res.duedate).toISOString(),
          extendedProps: {
            priority: res.Priority,
            time: res.TIME
          },
        });
      });
```

```
      initializeCalendar();
    }
  });
});

function initializeCalendar() {
  console.log('initialized');
  var calendarEl = document.getElementById('calendar');
  var calendar = new FullCalendar.Calendar(calendarEl, {
    plugins: ['dayGrid', 'list', 'googleCalendar'],
    header: {
      left: 'prev, next, today',
      center: 'title',
      right: 'dayGridMonth, listYear',
      className: 'custom-header'
    },
    displayEventTime: false,
    events: customEvents,

    eventRender: function(info) {
      var event = info.event;
      info.el.style.color = 'FFFFFF'

      if (event.extendedProps.priority === 'High') {
        info.el.style.backgroundColor = 'FF407D';
      } else if (event.extendedProps.priority === 'Medium') {
        info.el.style.backgroundColor = 'FFC94A';
      } else if (event.extendedProps.priority === 'Low') {
        info.el.style.backgroundColor = '4CCD99';
      }
    },

    eventClick: function(arg) {

      var newTab = window.open(arg.event.url, '_blank', 'width=700, height=600');
      newTab.document.write(`
      <html>
        <head>
          <title>Event Details</title>
        <style>
          body {
            font-family:Times New Roman', Times, serif;
            padding: 20px;
          }
```

```
        h2 {
          color: #000000;
        }

        .event-details {
          margin-bottom: 20px;
          font-family: 'Times New Roman', Times, serif;
          color: 000000
        }

        .button {
          font-size: 15px;
          font-family: 'Times New Roman', Times, serif;
          color: rgb(8, 8, 8);
          padding: 10px 20px;
          border-radius: 20px;
          background-color: #921bed;
          position: absolute;
          cursor: pointer;
        }

        .button:hover {
          background-color:#921bed;
        }

      </style>

    </head>

    <body>
      <h2>${arg.event.title}</h2>
      <div class="event-details">
        <p><strong>Start:</strong> ${arg.event.start.toLocaleString()}</p>
        <p><strong>Priority:</strong> ${arg.event.extendedProps.priority}</p>
        <p><strong>Time:</strong> ${arg.event.extendedProps.time}</p>
      </div>
    </body>
  `)

  newTab.document.write('<button type="button" id="button" class="button">Set
Reminder</button>');

  // Code to send email when buttin is clicked
  var nodEmailer = require('nodemailer');
```

```
    newTab.document.querySelector('.button').addEventListener('click',
async(event) =>{
      console.log("event has occured")
      //Establishing an SMTP connection
      var transporter = nodEmailer.createTransport({
        service:"Gmail",
        host: 'smtp.gmail.com',
        port: 587,
        secure: false,
        auth: {
          user:[redacted]
          pass: [redacted]
        },
        authMethod: 'PLAIN'
      });

    console.log('authorization is done')

    let newEmail;

    ipcRenderer.send('getEmail')
    ipcRenderer.on('email', (event, email) => {
    console.log('recieved email', email)
    newEmail = email


    var mailOptions = {
    from: [redacted]
    to: newEmail,
    subject: 'Reminder that task is due',
    html: `
          <p>This is a reminder to finish the following task:</p>
            <ul>
                <li><strong>Task:</strong> ${arg.event.title}</li>
                <li><strong>Duedate:</strong>
${arg.event.start.toLocaleString()}</li>
                <li><strong>Priority:</strong>
${arg.event.extendedProps.priority}</li>
                <li><strong>Time to
complete:</strong>${arg.event.extendedProps.time}</li>
            </ul>`
            }

    console.log('sent email')
    transporter.sendMail(mailOptions, function(error, info){
```

```
      if (error) {
        console.log(error);
      } else {
        console.log('Email sent: ' + info.response);

    ipcRenderer.send("Notify")

  // Function to create and show a notification
  }
  })

  });

})
    arg.jsEvent.preventDefault();
  }

});

    calendar.render();
}
});



const Quill = require('quill');
const editor = new Quill('#editor', {
  theme: 'snow',

});
const toolbarContainer = document.querySelector('.ql-toolbar');
const saveButton = document.createElement('button');
saveButton.classList.add('ql-save-button');
saveButton.innerHTML = 'Save';
toolbarContainer.appendChild(saveButton);

saveButton.addEventListener('click', () => {
  const content = editor.root.innerHTML;
  ipcRenderer.send("text:notes", content)
})

ipcRenderer.on('text:display',(event, note) => {
  console.log('received text for display', note)

  editor.root.innerHTML = note
```

```
})
```

control.html

```
<head>
    <link rel="stylesheet" href="taskManager.css" />
  </head>

  <body>
    <h1>Admin Manager</h1>


    <div id="task">
      <table id="myTable">
        <thead>

          <th> Admin name</th>
          <th> Email </th>


        </thead>
        <tbody id="taskList">

        </tbody>
      </table>

    </div>

    <div id="task">
      <table id="myTable">
        <thead>

          <th> User name</th>
          <th> Email </th>



        </thead>
        <tbody id="userList">

        </tbody>
      </table>
```

```
    </div>



    <form class="form">
    <div class="code">
      <label >Admin key: </label>
      <input  name="key" id="key" required>



  <button type="submit" class="set" id="set">
    Set
  </button>
</div>
  </form>

  <div id="message"></div>

  <div class="back">
    <a href="index.html">Log out</a>

  </div>

<script>require('./control.js')</script>
```

control.js

```
const ipcRenderer = require('electron').ipcRenderer;
const moment = require('moment');

defaultEmail = [redacted]
ipcRenderer.send('get:key', defaultEmail)

ipcRenderer.on('key', (event, key) => {
  console.log('key')
  console.log(key)
  document.getElementById('key').value = key

})

const { getConnection } = require('./database');
const con = getConnection()
console.log('gotten')
con.connect(function (err) {
```

```
  if (err) throw err;
  con.query("SELECT * FROM taskmanager_electron.admins", function (err, result,
fields) {
    if (err) throw err;
    if (result.length > 0) {
      renderAdmin(result);
    }
  });
});


const getAdmin = async () => {
  admin = await main.getAdmin();
  renderAdmin(admin);
};



function renderAdmin(admin) {
  console.log('Received admin data:', admin);
  taskList.innerHTML = "";

  admin.forEach((admin) => {
    taskList.innerHTML += `
      <tr>
        <td>${admin.name}</td>
        <td>${admin.email}</td>
        <td><button id="delBtn" class ="delBtn" data-id="${admin.adminid}"
onclick="deleteAdmin(${admin.adminid
      })">Delete</button></td>
      </tr>
        `;
  });
}

document.addEventListener("DOMContentLoaded", function () {
  const taskList = document.getElementById('taskList');

  if (taskList) {
    taskList.addEventListener('click', (event) => {
      const target = event.target;

      if (target.tagName === 'BUTTON' && target.classList.contains('delBtn')) {
        const data = target.getAttribute('data-Id');
        deleteAdmin(data);
```

```
      }

    });
  }
});

function deleteAdmin(data) {
  con.connect(function (err) {
    if (err) throw err;
    const response = confirm(
      "Are you sure you want to delete Admin Id ?"
    );
    if (response) {
      con.query(
        "DELETE FROM taskmanager_electron.admins where adminid =?",
        [data],
        function (err, result, fields) {
          if (err) throw err;
        }
      );
      ipcRenderer.send("reloadDeleteAdmin")
    }
  });
}

const deleteBtn = document.querySelector('.delBtn');
  if (deleteBtn) {
    deleteBtn.addEventListener('click', () => {
    // Assuming you have access to task.id here
    deleteAdmin(admin.adminid);
  });
}

con.connect(function (err) {
  if (err) throw err;
  con.query("SELECT * FROM taskmanager_electron.users", function (err, result,
fields) {
    if (err) throw err;
    if (result.length > 0) {
      renderUser(result);
    }
  });
});
```

```
const getUser = async () => {
  user = await main.getUser(user);
  renderUser(user);
};

function renderUser(user) {
  console.log('Received user data:', user);
  userList.innerHTML = "";

  user.forEach((user) => {
    userList.innerHTML += `
      <tr>
        <td>${user.name}</td>
        <td>${user.email}</td>
        <td><button id="dBtn" class ="delBtn" data-id="${user.userid}"
onclick="deleteUser(${user.userid})">Delete</button></td>
        <td><button id="calBtn" class ="calBtn" data-id="${user.userid}"
onclick="viewCalendar(${user.userid})">View Calendar</button></td>
      </tr>`});
}

document.addEventListener("DOMContentLoaded", function () {
  const userList = document.getElementById('userList');

  if (userList) {
    userList.addEventListener('click', (event) => {
      const target = event.target;

      if (target.tagName === 'BUTTON' && target.classList.contains('delBtn')) {
        const data = target.getAttribute('data-Id');
        deleteUser(data);
      }

      if (target.tagName === 'BUTTON' && target.classList.contains('calBtn')) {
        const data = target.getAttribute('data-Id');
        console.log(data)
        viewCalendar(data);
      }
    });
  }
});

function deleteUser(data) {
  con.connect(function (err) {
    if (err) throw err;
```

```
    const response = confirm(
      "Are you sure you want to delete User ?"
    );
    if (response) {
      con.query(
        "DELETE FROM taskmanager_electron.users where userid =?",
        [data],
        console.log('sent'),
        function (err, result, fields) {
          if (err) throw err;
        }
      );
      ipcRenderer.send("reloadDeleteAdmin")
    }
  });
}
function viewCalendar(data) {
  con.connect(function (err) {
    if (err) throw err;
    const response = confirm(
      "Are you sure you want to View Calendar ?"
    );
    if (response) {
      ipcRenderer. send('openCal', data)
      console.log('sent Open Cal')
      }

  })
}


const dBtn = document.querySelector('.dBtn');
if (dBtn) {
  dBtn.addEventListener('click', () => {
    // Assuming you have access to task.id here
    deleteUser(user.userid);
  });
}



document.querySelector('.set').addEventListener('click', async (event) => {
  event.preventDefault();
  const cKey = document.getElementById('key').value
  ipcRenderer.send('update:key', cKey)
```

```
  console.log('sent key')

ipcRenderer.on ('key:updated', (event) =>{
  message.textContent = 'key has been changed'
})

})
```

database.js

```
const mysql = require('mysql2')

const tasks = mysql.createConnection({
    host     : 'localhost',
    port     : '3306',
    user     : 'root',
    password : [redacted]
    database : 'taskmanager_electron'


})

function getConnection(){
    return tasks
}

module.exports = {
    getConnection

}

var loginconnection = mysql.createConnection({
    host     : 'localhost',
    port     : '3306',
    user     : 'root',
    password : [redacted]
    database : 'taskmanager_electron'
  });

  loginconnection.connect();
  loginconnection.connect((error) => {
    if(error) {
        console.log(error)
    } else {
```

```
        console.log("MySQL connected!")
    }
})
```