**Documentation: homeController.js**

This code file is located at src/controllers/homeController.js.

The purpose of this code file is to handle the routes for rendering various pages of the application.

**Functions**

**homePage()**

This function handles the request for the home page of the application.

*const* homePage = *async* (*req*, *res*) *=>* {

*const* [settings] = await connection.query('SELECT `app` FROM admin');

*let* app = settings[0].app;

    return *res*.render("home/index.ejs", { app});

}

**Response:**

* Render the home/index.ejs template.
* Pass the app variable to the template.

**activityPage()**

This function handles the request for the activity page.

*const* activityPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/activity.ejs");

}

**Response:**

* Render the checkIn/activity.ejs template.

**invitationRulesPage()**

This function handles the request for the invitation rules page.

*const* invitationRulesPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/invitationRules.ejs");

  };

**Response:**

* Render the checkIn/invitationRules.ejs template.

**rabateRatioPage()**

This function handles the request for the rebate ratio page.

*const* rabateRatioPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/rebateratio.ejs");

};

**Response:**

* Render the promotion/rebateratio.ejs template.

**rebatePage()**

This function handles the request for the rebate page.

*const* rebatePage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/rebate.ejs");

}

**Response:**

* Render the checkIn/rebate.ejs template.

**vipPage()**

This function handles the request for the VIP page.

*const* vipPage = *async* (*req*, *res*) *=>* {

*let* auth = *req*.cookies.auth;

*const* [userinfo] = await connection.query('SELECT `name\_user` FROM users WHERE `token` = ? ', [auth]);

*let* userid = userinfo[0].name\_user;

    return *res*.render("checkIn/vip.ejs", {  UserName : userid });

}

**Request:**

* auth: Token from the user's cookie.

**Response:**

* Fetch the user's name using the token from the database.
* Render the checkIn/vip.ejs template.
* Pass the UserName variable to the template.

**jackpotPage()**

This function handles the request for the jackpot page.

*const* jackpotPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/jackpot.ejs");

}

**Response:**

* Render the checkIn/jackpot.ejs template.

**dailytaskPage()**

This function handles the request for the daily task page.

*const* dailytaskPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/dailytask.ejs");

}

**Response:**

* Render the checkIn/dailytask.ejs template.

**invibonusPage()**

This function handles the request for the invitation bonus page.

*const* invibonusPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/invibonus.ejs");

}

**Response:**

* Render the checkIn/invibonus.ejs template.

**checkInPage()**

This function handles the request for the check-in page.

*const* checkInPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/checkIn.ejs");

}

**Response:**

* Render the checkIn/checkIn.ejs template.

**checkDes()**

This function handles the request for the check-in description page.

*const* checkDes = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/checkDes.ejs");

}

**Response:**

* Render the checkIn/checkDes.ejs template.

**checkRecord()**

This function handles the request for the check-in record page.

*const* checkRecord = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/checkRecord.ejs");

}

**Response:**

* Render the checkIn/checkRecord.ejs template.

**addBank()**

This function handles the request for the add bank page.

*const* addBank = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/addbank.ejs");

}

**Response:**

* Render the wallet/addbank.ejs template.

**promotionPage()**

This function handles the request for the promotion page.

*const* promotionPage = *async* (*req*, *res*) *=>* {

*let* auth = *req*.cookies.auth;

*const* [rows] = await connection.execute('SELECT `user\_level` FROM `users` WHERE `token` = ? AND veri = 1', [auth]);

*var* user\_level = rows[0].user\_level;

    return *res*.render("promotion/promotion.ejs", {user\_level});

}

**Request:**

* auth: Token from the user's cookie.

**Response:**

* Fetch the user's level using the token from the database.
* Render the promotion/promotion.ejs template.
* Pass the user\_level variable to the template.

**subordinatesPage()**

This function handles the request for the subordinates page.

*const* subordinatesPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/subordinates.ejs");

  };

**Response:**

* Render the promotion/subordinates.ejs template.

**promotion1Page()**

This function handles the request for the promotion 1 page.

*const* promotion1Page = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/promotion1.ejs");

}

**Response:**

* Render the promotion/promotion1.ejs template.

**promotionmyTeamPage()**

This function handles the request for the my team page of the promotion section.

*const* promotionmyTeamPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/myTeam.ejs");

}

**Response:**

* Render the promotion/myTeam.ejs template.

**promotionDesPage()**

This function handles the request for the promotion description page.

*const* promotionDesPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/promotionDes.ejs");

}

**Response:**

* Render the promotion/promotionDes.ejs template.

**comhistoryPage()**

This function handles the request for the commission history page.

*const* comhistoryPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/comhistory.ejs");

}

**Response:**

* Render the promotion/comhistory.ejs template.

**mybethistoryPage()**

This function handles the request for the my bet history page.

*const* mybethistoryPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/mybethistory.ejs");

}

**Response:**

* Render the promotion/mybethistory.ejs template.

**tutorialPage()**

This function handles the request for the tutorial page of the promotion section.

*const* tutorialPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/tutorial.ejs");

}

**Response:**

* Render the promotion/tutorial.ejs template.

**bonusRecordPage()**

This function handles the request for the bonus record page of the promotion section.

*const* bonusRecordPage = *async* (*req*, *res*) *=>* {

    return *res*.render("promotion/bonusrecord.ejs");

}

**Response:**

* Render the promotion/bonusrecord.ejs template.

**transactionhistoryPage()**

This function handles the request for the transaction history page of the wallet section.

*const* transactionhistoryPage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/transactionhistory.ejs");

}

**Response:**

* Render the wallet/transactionhistory.ejs template.

**walletPage()**

This function handles the request for the wallet page.

*const* walletPage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/index.ejs");

}

**Response:**

* Render the wallet/index.ejs template.

**rechargePage()**

This function handles the request for the recharge page of the wallet section.

*const* rechargePage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/recharge.ejs", {

        MinimumMoney: process.env.MINIMUM\_MONEY

    });

}

**Response:**

* Render the wallet/recharge.ejs template.
* Pass the MinimumMoney variable to the template.

**rechargeAwardCollectionRecord()**

This function handles the request for the recharge award collection record page.

*const* rechargeAwardCollectionRecord = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/rechargeAwardCollectionRecord.ejs");

  };

**Response:**

* Render the checkIn/rechargeAwardCollectionRecord.ejs template.

**rechargerecordPage()**

This function handles the request for the recharge record page of the wallet section.

*const* rechargerecordPage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/rechargerecord.ejs");

}

**Response:**

* Render the wallet/rechargerecord.ejs template.

**withdrawalPage()**

This function handles the request for the withdrawal page of the wallet section.

*const* withdrawalPage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/withdrawal.ejs");

}

**Response:**

* Render the wallet/withdrawal.ejs template.

**withdrawalrecordPage()**

This function handles the request for the withdrawal record page of the wallet section.

*const* withdrawalrecordPage = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/withdrawalrecord.ejs");

}

**Response:**

* Render the wallet/withdrawalrecord.ejs template.

**transfer()**

This function handles the request for the transfer page of the wallet section.

*const* transfer = *async* (*req*, *res*) *=>* {

    return *res*.render("wallet/transfer.ejs");

}

**Response:**

* Render the wallet/transfer.ejs template.

**mianPage()**

This function handles the request for the main page of the member section.

**Request:**

* auth: Token from the user's cookie.

**Response:**

* Fetch the user's level and the customer support contact from the database.
* Render the member/index.ejs template.
* Pass the level and cskh variables to the template.

*const* mianPage = *async* (*req*, *res*) *=>* {

*let* auth = *req*.cookies.auth;

*const* [user] = await connection.query('SELECT `level` FROM users WHERE `token` = ? ', [auth]);

*const* [settings] = await connection.query('SELECT `cskh` FROM admin');

*let* cskh = settings[0].cskh;

*let* level = user[0].level;

    return *res*.render("member/index.ejs", { level, cskh });

}

In the given code snippet:

1. A function mainPage is defined as an asynchronous function that
2. takes req and res as parameters.
3. The function retrieves the auth value from the cookies in req.
4. It then queries the database to fetch the level of the user based on the provided auth token.
5. Similarly, it queries the database to fetch the cskh setting from the admin table.
6. The retrieved values are then used to set cskh and level variables accordingly.
7. Finally, the function renders an EJS template member/index.ejs and passes the level and cskh values to it for rendering.

**safePage()**

This function handles the request for the safe page of the member section.

**Response:**

* Fetch the sandbox mode and stake ROI values from the environment variables.
* Render the member/safe.ejs template.
* Pass the sandbox, roi1, roi2, roi3, and roi4 variables to the template.

*const* safePage = *async* (*req*, *res*) *=>* {

*var* sandbox = process.env.SANDBOX\_MODE;

*var* roi1 = process.env.STAKE\_ROI\_1;

*var* roi2 = process.env.STAKE\_ROI\_3;

*var* roi3 = process.env.STAKE\_ROI\_6;

*var* roi4 = process.env.STAKE\_ROI\_12;

    return *res*.render("member/safe.ejs",{ sandbox,roi1,roi2,roi3,roi4});

}

In this code snippet, an asynchronous arrow function safePage is defined that takes req and res as parameters.

1. Inside the function, values are assigned to variables sandbox, roi1, roi2, roi3, and roi4 by accessing the corresponding environment variables using process.env.
2. Finally, the function returns the rendered template safe.ejs with the variables sandbox, roi1, roi2, roi3, and roi4 passed as data to the template.

**languegePage()**

This function handles the request for the language page of the member section.

*const* languegePage = *async* (*req*, *res*) *=>* {

*let* lang = *req*.cookies.lang;

    return *res*.render("member/language.ejs",{lang});

}

**Request:**

* lang: Language from the user's cookie.

**Response:**

* Render the member/language.ejs template.
* Pass the lang variable to the template.

**avatarpage()**

This function handles the request for the avatar page of the member section.

*const* avatarpage = *async* (*req*, *res*) *=>* {

*let* lang = *req*.cookies.lang;

    return *res*.render("member/avatar.ejs");

}

**Response:**

* Render the member/avatar.ejs template.

**d\_get\_betting()**

This function handles the request for the betting details of a particular game.

**Request:**

* auth: Token from the user's cookie.
* gameJoin: Game name to fetch the betting details.

**Response:**

* Fetch the betting details from the database based on the game name and user's phone number.
* Return a JSON response with the message, status, and betting details.

*const* d\_get\_betting = *async* (*req*, *res*) *=>* {

*let* auth = *req*.cookies.auth;

*const* [user] = await connection.query('SELECT `phone` FROM users WHERE `token` = ? ', [auth]);

*let* phone = user[0].phone;

*let* gameJoin = *req*.body.gameJoin;

*var* betting\_list = '';

    if( gameJoin == "WinGo")

    {

        [betting\_list] = await connection.query('SELECT \* FROM minutes\_1 WHERE `phone` = ? AND status NOT IN ( 0 )  ORDER BY `id` DESC', [phone]);

    }

    else if(gameJoin == "5D")

    {

        [betting\_list] = await connection.query('SELECT \* FROM result\_5d WHERE `phone` = ? AND status NOT IN ( 0 )  ORDER BY `id` DESC ', [phone]);

    }

    else if(gameJoin == "K3")

    {

        [betting\_list] = await connection.query('SELECT \* FROM result\_k3 WHERE `phone` = ? AND status NOT IN ( 0 )  ORDER BY `id` DESC ', [phone]);

    }

    else if(gameJoin == "Trx Wingo")

        {

            [betting\_list] = await connection.query('SELECT \* FROM trx\_wingo\_bets WHERE `phone` = ? AND status NOT IN ( 0 )  ORDER BY `id` DESC ', [phone]);

        }

    return *res*.status(200).json({

        message: 'Success',

        status: true,

        datas: betting\_list,

    });

}

This code defines an asynchronous function d\_get\_betting that takes req and res as parameters. It first retrieves the auth token from cookies and then queries the database to get the user's phone number based on the token.

1. It then checks the value of gameJoin from the request body and performs different queries based on the value. If gameJoin is 'WinGo', '5D', 'K3', or 'Trx Wingo', corresponding database queries are executed to fetch betting information for the user.
2. The function finally returns a JSON response with a status code of 200, along with a message indicating success, a boolean status value, and the betting\_list obtained from the database query.

**aboutPage()**

This function handles the request for the about page of the member section.

*const* aboutPage = *async* (*req*, *res*) *=>* {

    return *res*.render("member/about/index.ejs");

}

**Response:**

* Render the member/about/index.ejs template.

**notificationPage()**

This function handles the request for the notification page of the member section.

*const* notificationPage = *async* (*req*, *res*) *=>* {

    return *res*.render("member/notification.ejs");

}

**Response:**

* Render the member/notification.ejs template.

**wingochat()**

This function handles the request for the wingo chat page of the member section.

**Response:**

* Fetch the latest wingo details from the database.
* Render the member/wingochat.ejs template.
* Pass the d\_period, d\_amount, and Firebase credentials variables to the template.

*const* wingochat = *async* (*req*, *res*) *=>* {

*const* [winGo1] = await connection.execute('SELECT \* FROM `wingo` WHERE `game` = "wingo" ORDER BY `id` DESC LIMIT 2 ', []);

*const* period = winGo1[1].period;

*const* amount = winGo1[1].amount;

*var* f\_api= process.env.Firebase\_Apikey;

*var* f\_authdomain= process.env.Firebase\_AuthDomain;

*var* f\_dburl= process.env.Firebase\_Dburl;

*var* f\_projid= process.env.Firebase\_ProjId;

*var* f\_stobck= process.env.Firebase\_StorageBucket;

*var* f\_messId= process.env.Firebase\_MessageSenId;

*var* f\_appid= process.env.Firebase\_AppId;

*var* f\_mesuareId= process.env.Firebase\_MeasurementId;

    return *res*.render("member/wingochat.ejs", { d\_period: period, d\_amount :amount,d\_f\_api: f\_api, d\_f\_authdomain :f\_authdomain,d\_f\_dburl:f\_dburl,d\_f\_projid:f\_projid,d\_f\_stobck:f\_stobck,d\_f\_messId:f\_messId,d\_f\_appid:f\_appid,d\_f\_mesuareId:f\_mesuareId});

}

In this code snippet:

1. A function named wingochat is declared as an asynchronous function that takes two parameters: req and res.
2. The function queries a database table named wingo to retrieve the latest 2 records where the game column is 'wingo' and orders them by id.
3. The values of period and amount are extracted from the second record of the result.
4. Environment variables related to Firebase are stored in variables for later use.
5. Finally, the function renders an EJS template named wingochat.ejs and passes the extracted data (period, amount, Firebase variables) as an object to the template.

**k3chat()**

This function handles the request for the k3 chat page of the member section.

**Response:**

* Fetch the latest K3 details from the database.
* Render the member/k3chat.ejs template.
* Pass the kd\_period, kd\_amount, and Firebase credentials variables to the template.

*const* k3chat = *async* (*req*, *res*) *=>* {

*const* [k31] = await connection.execute('SELECT \* FROM `k3` WHERE `game` = "1" ORDER BY `id` DESC LIMIT 2 ', []);

*const* k\_period = k31[1].period;

*const* k\_amount = k31[1].result;

*var* f\_api= process.env.Firebase\_Apikey;

*var* f\_authdomain= process.env.Firebase\_AuthDomain;

*var* f\_dburl= process.env.Firebase\_Dburl;

*var* f\_projid= process.env.Firebase\_ProjId;

*var* f\_stobck= process.env.Firebase\_StorageBucket;

*var* f\_messId= process.env.Firebase\_MessageSenId;

*var* f\_appid= process.env.Firebase\_AppId;

*var* f\_mesuareId= process.env.Firebase\_MeasurementId;

    return *res*.render("member/k3chat.ejs", { kd\_period: k\_period, kd\_amount :k\_amount,d\_f\_api: f\_api, d\_f\_authdomain :f\_authdomain,d\_f\_dburl:f\_dburl,d\_f\_projid:f\_projid,d\_f\_stobck:f\_stobck,d\_f\_messId:f\_messId,d\_f\_appid:f\_appid,d\_f\_mesuareId:f\_mesuareId});

}

The provided code defines an asynchronous function k3chat that takes req and res as parameters.

1. It executes a SQL query to fetch data from the 'k3' table based on the game condition and orders it by 'id' in descending order, limiting to 2 results.
2. It then extracts the 'period' and 'result' values from the second element of the fetched data (k31[1]).
3. Following this, it retrieves various Firebase environment variables using process.env and stores them in corresponding variables.
4. Finally, it renders a template 'k3chat.ejs' using res.render and passes the extracted data and Firebase variables as an object to the template.

**d5chat()**

This function handles the request for the 5D chat page of the member section.

**Response:**

* Fetch the latest 5D details from the database.
* Render the member/d5chat.ejs template.
* Pass the d5\_period, d5\_amount, and Firebase credentials variables to the template.

*const* d5chat = *async* (*req*, *res*) *=>* {

*const* [d51] = await connection.execute('SELECT \* FROM `d5` WHERE `game` = "1" ORDER BY `id` DESC LIMIT 2 ', []);

*const* d5\_period = d51[1].period;

*const* d5\_amount = d51[1].result;

*var* f\_api= process.env.Firebase\_Apikey;

*var* f\_authdomain= process.env.Firebase\_AuthDomain;

*var* f\_dburl= process.env.Firebase\_Dburl;

*var* f\_projid= process.env.Firebase\_ProjId;

*var* f\_stobck= process.env.Firebase\_StorageBucket;

*var* f\_messId= process.env.Firebase\_MessageSenId;

*var* f\_appid= process.env.Firebase\_AppId;

*var* f\_mesuareId= process.env.Firebase\_MeasurementId;

    return *res*.render("member/d5chat.ejs", { d5\_period: d5\_period, d5\_amount :d5\_amount , d\_f\_api: f\_api, d\_f\_authdomain :f\_authdomain,d\_f\_dburl:f\_dburl,d\_f\_projid:f\_projid,d\_f\_stobck:f\_stobck,d\_f\_messId:f\_messId,d\_f\_appid:f\_appid,d\_f\_mesuareId:f\_mesuareId});

}

The given code defines an asynchronous function d5chat that takes req and res as parameters.

1. Inside the function, a SQL query is executed to select data from a table named d5 where the column game equals 1.
2. The result is stored in d51, which is an array.
3. Subsequently, d5\_period and d5\_amount are extracted from the first element of d51.
4. Environment variables related to Firebase are accessed using process.env and stored in respective variables.
5. These variables hold values like API key, authentication domain, database URL, project ID, storage bucket, message sender ID, app ID, and measurement ID.
6. Finally, the function renders a template named d5chat.ejs and passes the extracted data along with Firebase variables as an object for rendering the view.

**recordsalary()**

This function handles the request for the salary record page of the member section.

*const* recordsalary = *async* (*req*, *res*) *=>* {

    return *res*.render("member/about/recordsalary.ejs");

}

**Response:**

* Render the member/about/recordsalary.ejs template.

**gameStatisticsPage()**

This function handles the request for the game statistics page of the member section.

*const* gameStatisticsPage = *async* (*req*, *res*) *=>* {

    return *res*.render("member/game\_statistics.ejs");

}

**Response:**

* Render the member/game\_statistics.ejs template.

**privacyPolicy()**

This function handles the request for the privacy policy page of the member section.

*const* privacyPolicy = *async* (*req*, *res*) *=>* {

    return *res*.render("member/about/privacyPolicy.ejs");

}

**Response:**

* Render the member/about/privacyPolicy.ejs template.

**newtutorial()**

This function handles the request for the new tutorial page of the member section.

*const* newtutorial = *async* (*req*, *res*) *=>* {

    return *res*.render("member/newtutorial.ejs");

}

**Response:**

* Render the member/newtutorial.ejs template.

**forgot()**

This function handles the request for the forgot password page of the member section.

**Request:**

* auth: Token from the user's cookie.

**Response:**

* Fetch the time OTP from the database based on the token.
* Render the member/forgot.ejs template.
* Pass the time variable to the template.

*const* forgot = *async* (*req*, *res*) *=>* {

*let* auth = *req*.cookies.auth;

*const* [user] = await connection.query('SELECT `time\_otp` FROM users WHERE token = ? ', [auth]);

*let* time = user[0].time\_otp;

    return *res*.render("member/forgot.ejs", { time });

}

The given code defines an asynchronous function forgot that takes req and res as parameters.

1. It reads the value of auth from the cookies in the request object. It then queries the database using connection.query to select the time\_otp from the users table where the token matches the auth value.
2. Once the query is executed, it extracts the time\_otp value from the result and assigns it to the variable time.
3. Finally, it renders the 'member/forgot.ejs' template passing the time value as data to be used in the template.

**redenvelopes()**

This function handles the request for the red envelopes page of the member section.

*const* redenvelopes = *async* (*req*, *res*) *=>* {

    return *res*.render("member/redenvelopes.ejs");

}

**Response:**

* Render the member/redenvelopes.ejs template.

**riskAgreement()**

This function handles the request for the risk agreement page of the member section.

*const* riskAgreement = *async* (*req*, *res*) *=>* {

    return *res*.render("member/about/riskAgreement.ejs");

}

**Response:**

* Render the member/about/riskAgreement.ejs template.

**myProfilePage()**

This function handles the request for the my profile page of the member section.

*const* myProfilePage = *async* (*req*, *res*) *=>* {

    return *res*.render("member/myProfile.ejs");

}

**Response:**

* Render the member/myProfile.ejs template.

**getSalaryRecord()**

This function handles the request for the salary record details of the current user.

*const* getSalaryRecord = *async* (*req*, *res*) *=>* {

*const* auth = *req*.cookies.auth;

*const* [rows] = await connection.query(`SELECT \* FROM users WHERE token = ?`, [auth]);

*let* rowstr = rows[0];

    if (!rows) {

        return *res*.status(200).json({

            message: 'Failed',

            status: false,

        });

    }

*const* [getPhone] = await connection.query(

        `SELECT \* FROM salary WHERE phone = ? ORDER BY time DESC`,

        [rowstr.phone]

    );

    console.log("asdasdasd : " + [rows.phone])

    return *res*.status(200).json({

        message: 'Success',

        status: true,

        data: {

        },

        rows: getPhone,

    })

}

The given code defines an asynchronous function getSalaryRecord that takes req and res as parameters.

1. It extracts auth from cookies in the request.
2. Then, it queries the database table 'users' to fetch records based on the token value in auth.
3. If no records are found, it returns a response with status 200 and a JSON object containing a message 'Failed' and status false.
4. Next, it queries the 'salary' table based on the phone number obtained from the previous query result.
5. The results are ordered by time in descending order.
6. A log statement is written to the console with the phone number from the retrieved row.
7. Finally, a success response with status true and the fetched records from the 'salary' table is sent in a JSON object.

**Request:**

* auth: Token from the user's cookie.

**Response:**

* Fetch the user's details from the database based on the token.
* Fetch the salary records from the database based on the user's phone number.
* Return a JSON response with the message, status, user's details, and salary records.

**attendancePage()**

This function handles the request for the attendance page.

*const* attendancePage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/attendance.ejs");

  };

**Response:**

* Render the checkIn/attendance.ejs template.

**attendanceRecordPage()**

This function handles the request for the attendance record page.

*const* attendanceRecordPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/attendanceRecord.ejs");

  };

**Response:**

* Render the checkIn/attendanceRecord.ejs template.

**attendanceRulesPage()**

This function handles the request for the attendance rules page.

*const* attendanceRulesPage = *async* (*req*, *res*) *=>* {

    return *res*.render("checkIn/attendanceRules.ejs");

  };

**Response:**

* Render the checkIn/attendanceRules.ejs template.