Saint Louis University
School of Accountancy, Management,
Computing and Information Studies

WEB TECHNOLOGIES IT312 - 9467
Finals Activity: Server-Side Web Scripting

SUBMITTED TO:
MA'AM. MA. CONCEPCION CLEMENTE
MA'AM BRITANNY BALDOVINO

TEAM MEMBERS:
BANAÑA, KEENRICK
De Castillo, Archeil
MOLINA, GABRIEL
RAMOS, JERICO
SAMILIN, MARK
YUSAY, AARON

Project Files

Authentication	
login.js	The login.js file exports a function that handles user login. It checks for the presence of a username and password, queries the database to verify the credentials, and generates a JSON Web Token (JWT) for authentication. If the login is successful, a cookie named studentUser for student session handling and custodianUser for custodian session handling, containing the JWT is set, and a success response is sent in JSON format. If the credentials are incorrect, an error response is sent.
userAuth.js	The file contains codes about setting up routes for user authentication using Express.js. It initializes an Express router and imports two middleware functions, login and register, from separate files. The router.post('/register', register) line defines a POST route for user registration, where requests to /register will be handled by the register middleware function. Similarly, router.post('/login', login) sets up a POST route for user login, directing requests to /login to the login middleware function.
loginFrontend.js	This JavaScript code handles form submission for user login. It captures the username and password, sends them to the server via a POST request to /api/login, and handles the response accordingly. If there's an error, it displays

	it to the user; otherwise, it redirects based on user type. Errors are logged to the console.

Custodian Module (Node.js)	
loggedinCustodian.js	This JavaScript middleware verifies if a custodian user is logged in by checking for a specific cookie (req.cookies.custodianUser). If the cookie is present, it decodes the JWT token contained within it using a secret key. Then, it queries the database to retrieve the user information based on the decoded user ID. If successful, it attaches the user information to the request object (req.custodianUser) and proceeds to the next middleware. If any errors occur during this process, they are logged, and the middleware proceeds to the next handler.
logoutCustodian.js	This JavaScript function handles the logout process for custodian users. It clears the 'custodianUser' cookie from the response object, effectively logging out the user. Then, it redirects the user to the '/login' page.
custodianCRUD.js	Manages the CRUD operations for custodian-related tasks.
custodianInventory.js	This file provides a class that encapsulates database operations related to a custodian's inventory, including retrieving data, inserting new items, toggling item availability, and deleting items with associated rent requests.
inventory.js	JavaScript file for handling frontend logic

	on custodian module.
custodianHome.html	Custodian Dashboard UI
custodian-style.css	CSS for styling the custodianHome.html

Student Module (Node.js)	
register.js	The register.js file exports a function that handles user registration. It checks for the presence of required fields (username, password, and fullname), performs a query to check if the username is already registered, and inserts a new user into the database if the username is available. The response is then sent in JSON format, indicating success or failure.
loggedin.js	The loggedIn.js file exports a middleware function, loggedIn, that checks for a 'userRegistered' cookie. If the cookie exists, it decodes the JWT, queries the database for user information, and attaches it to the request (req.user). Any errors are logged, and the middleware proceeds to the next function.
logout.js	The logout.js file exports a function named logout that clears the 'userRegistered' cookie and redirects the user to the '/login' route.
rentRequest.js	The rentRequest.js file defines an Express router handling student-related routes for managing inventory, rent requests, reservations, and returns. Key

	functionalities include retrieving available items, creating rent requests, managing pending requests, handling reservations, and processing returns. The routes interact with a database, querying and updating tables such as 'inventory,' 'rent_requests,' and 'reservations.' The responses are in JSON format, indicating success or failure for each operation.
register.js (Front-end)	The register.js file handles the submission of a registration form. It sends a POST request to '/api/register', processes the response, and updates the display based on success or error messages.
rentRequestFrontEnd.js	The rentRequestFrontEnd.js file handles the front-end logic for displaying student inventory, rent requests, and reservations. It makes asynchronous requests to the server API, dynamically loads tables with fetched data, and includes functions to handle rent requests, cancel requests, and returns. The script is designed to interact with a student-related web page, dynamically updating content based on the user's interactions.
studentHome.ejs	The studentHome.ejs file is an HTML template for a student dashboard. It includes navigation links, sections for displaying available inventory, pending requests, reservations, and user profile information. The content is conditionally

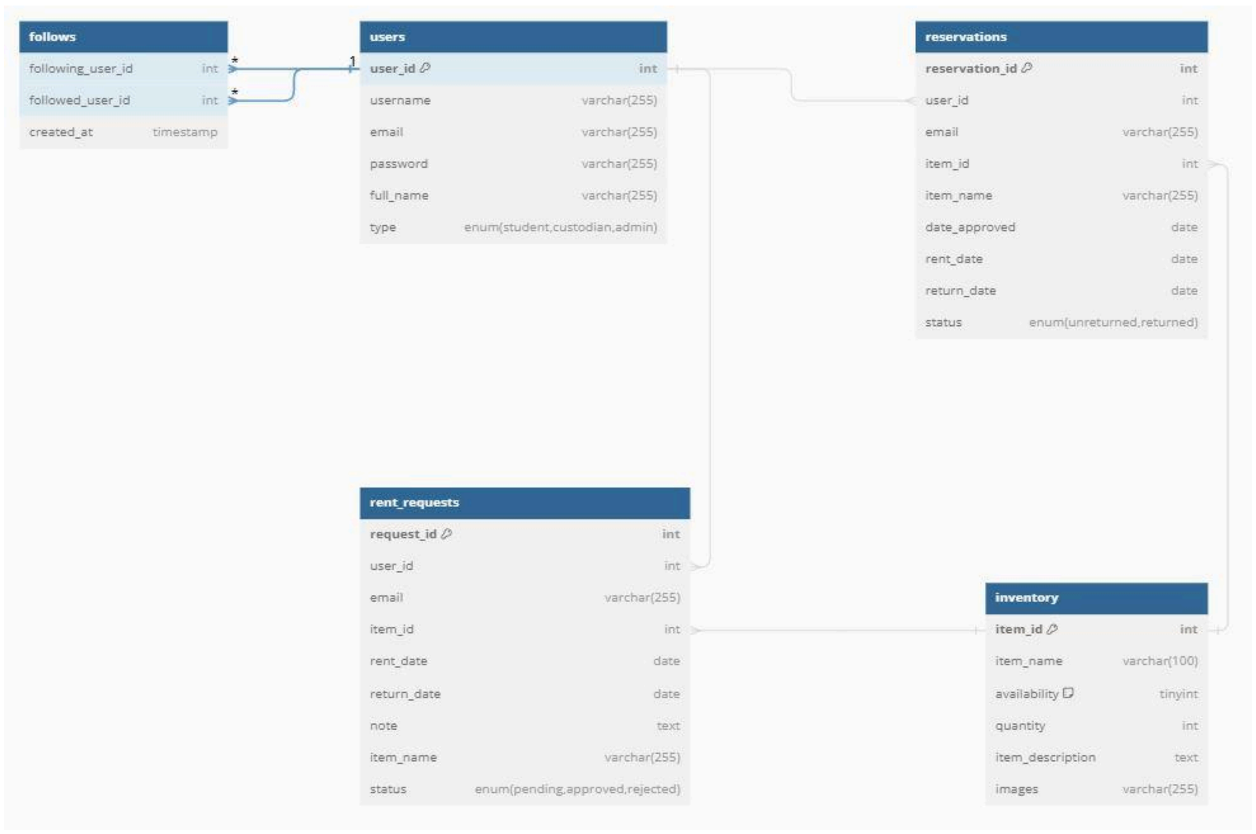
	displayed based on the user's login status. Additionally, the script at the end of the file links to the rentRequestFrontEnd.js file, which handles front-end logic for fetching and displaying data related to inventory, requests, and reservations.
student-style.css/Login.css/register.css	CSS for styling the studentHome.ejs/Login and Register for student module.

Node.js files	
app.js	Main application file for server.
db-config.js	Configuration file for connecting to the MySQL database.
pages.js	Handles routing for different pages.
package.json	Configuration and dependencies for the Node.js application

Web App Admin (PHP)	
index.php	The index.php file is a PHP script that fetches user accounts from a database, displays them in an HTML table, and provides a delete button for each user. Additionally, there's a JavaScript function deleteUser that sends an AJAX request to delete a user when the corresponding delete button is clicked.
delete_user.php	The delete_user.php file is a PHP script designed to handle the deletion of a user from a database. It connects to the

	database, retrieves the user_id from the GET parameters, and executes a SQL query to delete the corresponding user. The script then echoes a success message or an error message based on the deletion result.
admin-style.css	CSS styling for index.php

Relational Database Schema



Essential Features and Functionality

Custodian Functionality	
Login	To login an existing account.
Logout	To logout on the web page.

Add Item in inventory	To add an item or room in the inventory.
Update item quantity	To change the quantity of an item and marking it unavailable when the quantity is set to zero.
Delete Item in inventory	To delete an item in the inventory.
Approve	To approve a rent request from the student.
Reject	To reject a rent request from the student.
Navigations	To navigate to each section of the custodian dashboard web page.

Student Functionality	
Register	To register an account
Login	To login a registered account.
Logout	To logout on the web page.
Home	To go to the student dashboard.
Request Rent	To make a rent request to the custodian.
Cancel Request	To cancel and delete the rent request.
Return	To return the approved reserved item/room.
Navigations	To navigate to each section of the student dashboard web page.

Web App Admin	
Delete	To delete an existing account of a student in the users database.

Project Improvement Documentation

- Improve the UI of Login and Register Websites.
- Improved the form for registration, like adding email, separating the full name into first name and last name.
- Improve the UI of Student and Custodian Dashboard.
- Added Notes in Student Dashboard when requesting an item or a room, for special request to custodian.
- Added image upload when adding a room or item in Custodian Dashboard.
- Added a new type in the users table in the database for admin for authentication purposes.
- Added login functionality for admin.