

Program 4: Multitier Server Application

Documentation:

An Employee Management System is designed to simplify the process of record maintenance of employees in an organization. It helps in managing the information of employees for HR functions. In general, EMS is a part of a comprehensive Human Resource Management System. Most modern companies use computers to collect this information, making their system more accurate and problem-free. A database is often used to collect the information required.

Project structure:

I have created Java Enterprise web application “EMSystem”. Under the war folder, I have added the source packages as follows:

com.java.Entity

User.java : This entity class creates an user for HR of a company to login into the application with attributes ‘email’ and ‘password’ and getter, setter methods to access the attributes.

Employee.java : This entity class creates an employee works in a company that the HR’s can access the information using the application. Class attributes are eid, name, email, dob, gender, homeAddress, phoneNo, department, salary, designation, status and getter, setter methods to access the attributes.

com.java.Servlet

LoginServlet.java : This servlet will take the request parameter entered in the login.jsp and calls the UserBean “login” method to verify the valid user by passing the username and password. If it returns true, then its response is the next page “home.jsp”.

CreateServlet.java : This servlet takes the request parameter entered by the user to the EmployeeBean i.e. employee details. It calls the “createEmployee” method to create a new employee. If it returns true, then its response the next page “createResult.jsp”.

SearchServlet.java : This servlet takes the request parameter entered by the user to the EmployeeBean i.e. employee Id. It calls the “getEmployeeDetails” method to get a details of employee. If it returns true, then its response the next page “searchResult.jsp”.

UpdateServlet.java : This servlet takes the request parameter entered by the user to the EmployeeBean i.e. employee details. It calls the “updateEmployee” method to update existing employee details. If it returns true, then its response the next page “updateResult.jsp”.

DeleteServlet.java : This servlet takes the request parameter entered by the user to the EmployeeBean i.e. employee details. It calls the “deleteEmployee” method to delete the employee. If it returns true, then its response the next page “deleteResult.jsp”.

com.java.Bean

UserBeanLocal.java : This is the session bean interface created for UserBean where “login” method is declared.

UserBean.java : It stores all the business logic for the user. “Login” method is defined where it connects to database (third tier) through JDBC protocol. It then request parameters with the actual db values. If it matches, it returns true.

EmployeeBeanLocal.java : This is the session bean interface created for EmployeeBean where “createEmployee(Employee employee)”, “getEmployeeDetails(Employee employee)”, “updateEmployee(Employee employee)”, “deleteEmployee(Employee employee)” method is declared.

EmployeeBean.java : It stores all the business logic for the employee.

- “createEmployee” method where it connects to database (third tier) through JDBC protocol. It then executes insert query to DB with request parameters. If it executes, it returns true.
- “searchEmployee” method where it connects to database (third tier) through JDBC protocol. It then executes select query to DB with request parameters and stores the results in array list of strings. If it executes, it returns the list to the respective servlet.
- “updateEmployee” method where it connects to database (third tier) through JDBC protocol. It then executes update query to DB with request parameters. If it executes, it returns true.
- “deleteEmployee” method where it connects to database (third tier) through JDBC protocol. It then executes delete query to DB with request parameters. If it executes, it returns true.

.JSP pages under Web pages:

Login.jsp : login page for user to enter username and password

home.jsp : home page for HR dashboard to perform any of the function

createEmployee.jsp : page to create an employee and enter all the employee details

createResult.jsp : result page of create employee

searchEmployee.jsp : page to search an employee and enter the employee id

searchResult.jsp : result page of search employee

deleteEmployee.jsp : page to delete an employee and enter the employee id

deleteResult.jsp : result page of delete employee

updateEmployee.jsp : page to update an employee and enter the employee details to update

updateResult.jsp : result page of update employee

web.xml under WEB-INF: It contains all the entries of beans used and starts page “Login.jsp”

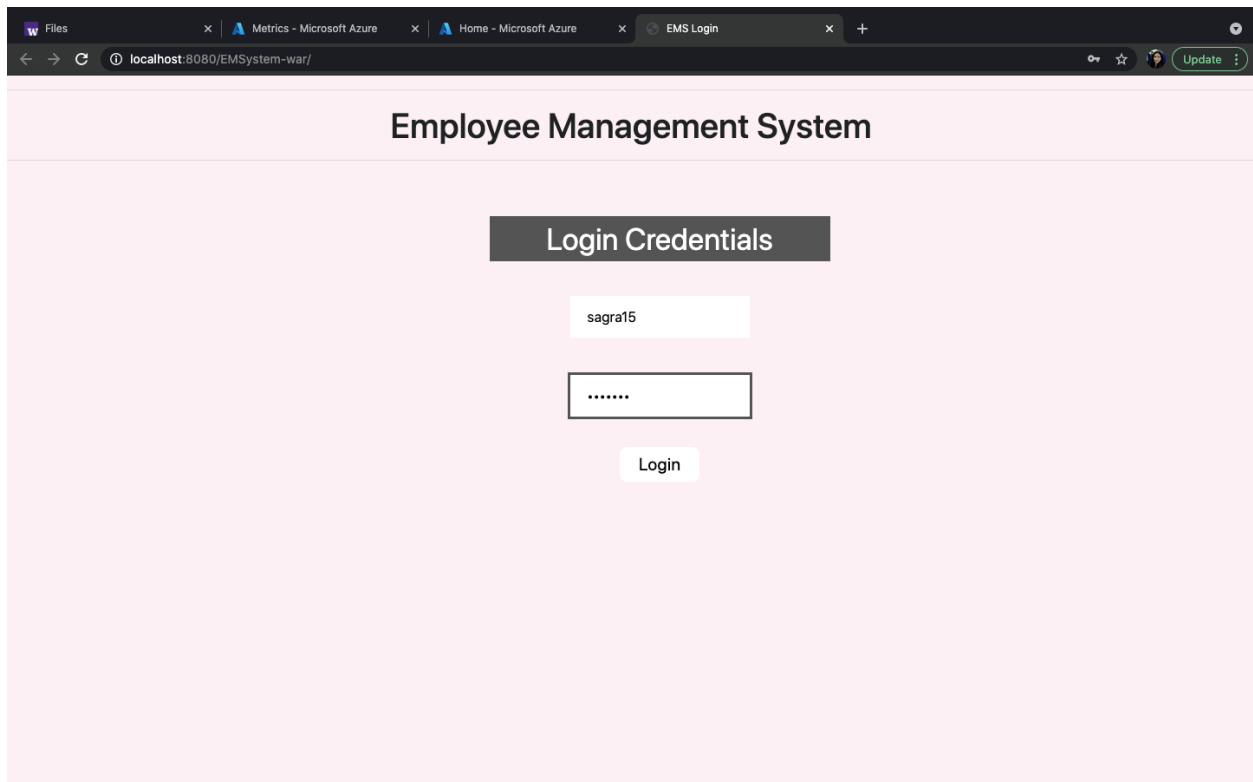
Source code:

Zip folder contains source code for:

1. Prog4
2. Lab8
3. Lab9

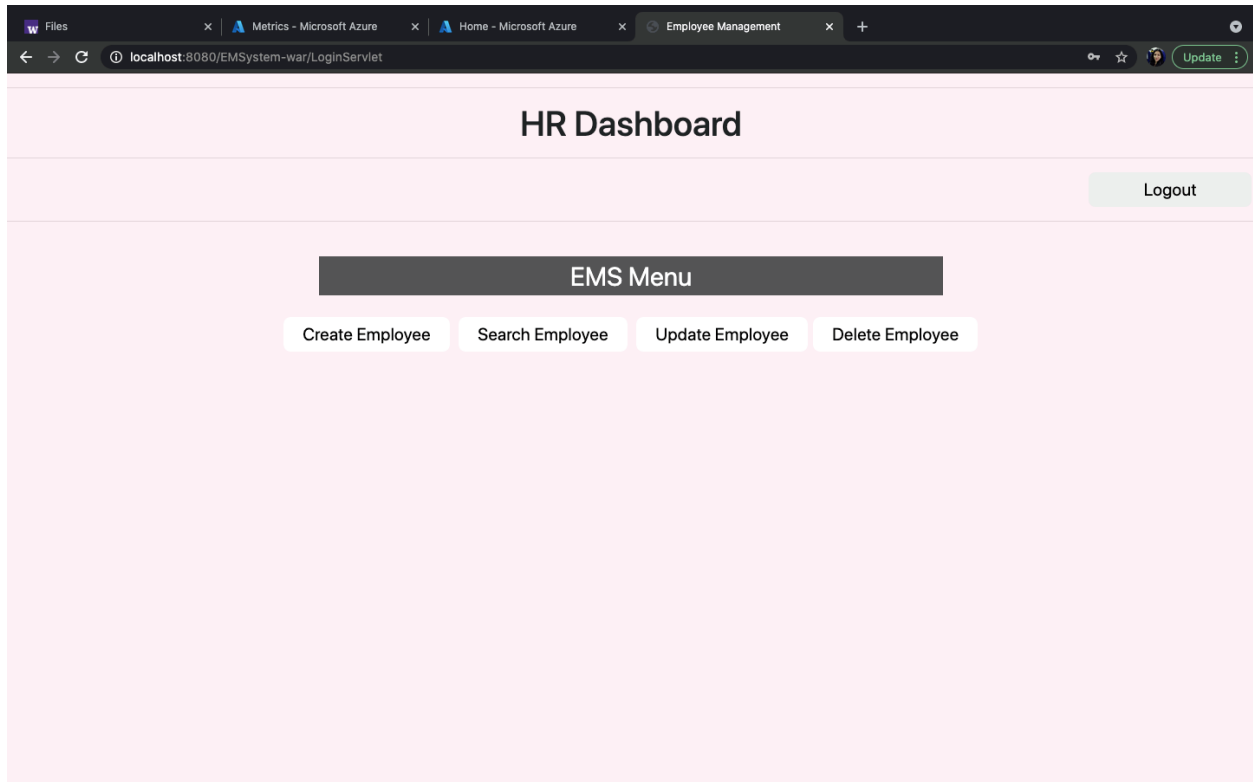
Execution output:

Login page of Employee Mangement System application

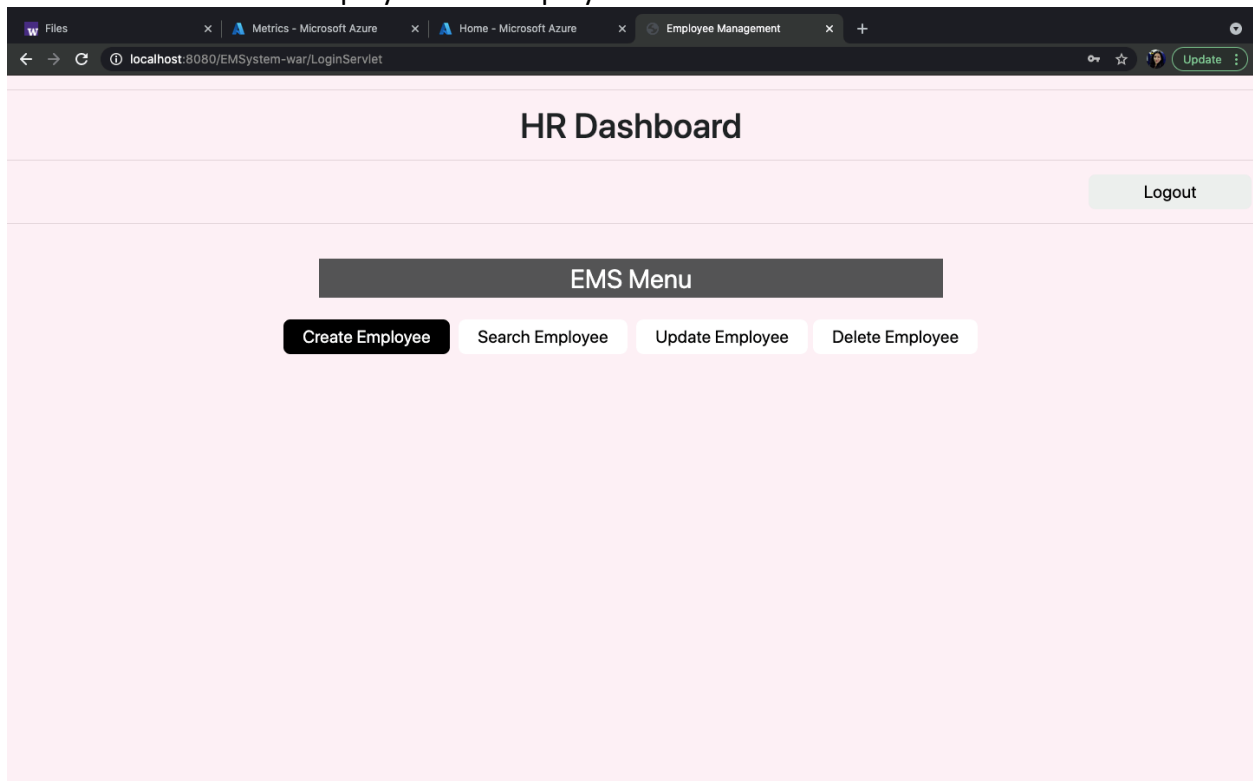


The screenshot shows a web browser window with the address bar displaying 'localhost:8080/EMSystem-war/'. The browser has several tabs open: 'Files', 'Metrics - Microsoft Azure', 'Home - Microsoft Azure', and 'EMS Login'. The page title is 'Employee Management System'. The main content area has a light pink background. In the center, there is a dark gray box labeled 'Login Credentials'. Below this box, there are two input fields: the first contains the text 'sagra15', and the second contains a series of dots '.....'. Below the input fields is a button labeled 'Login'.

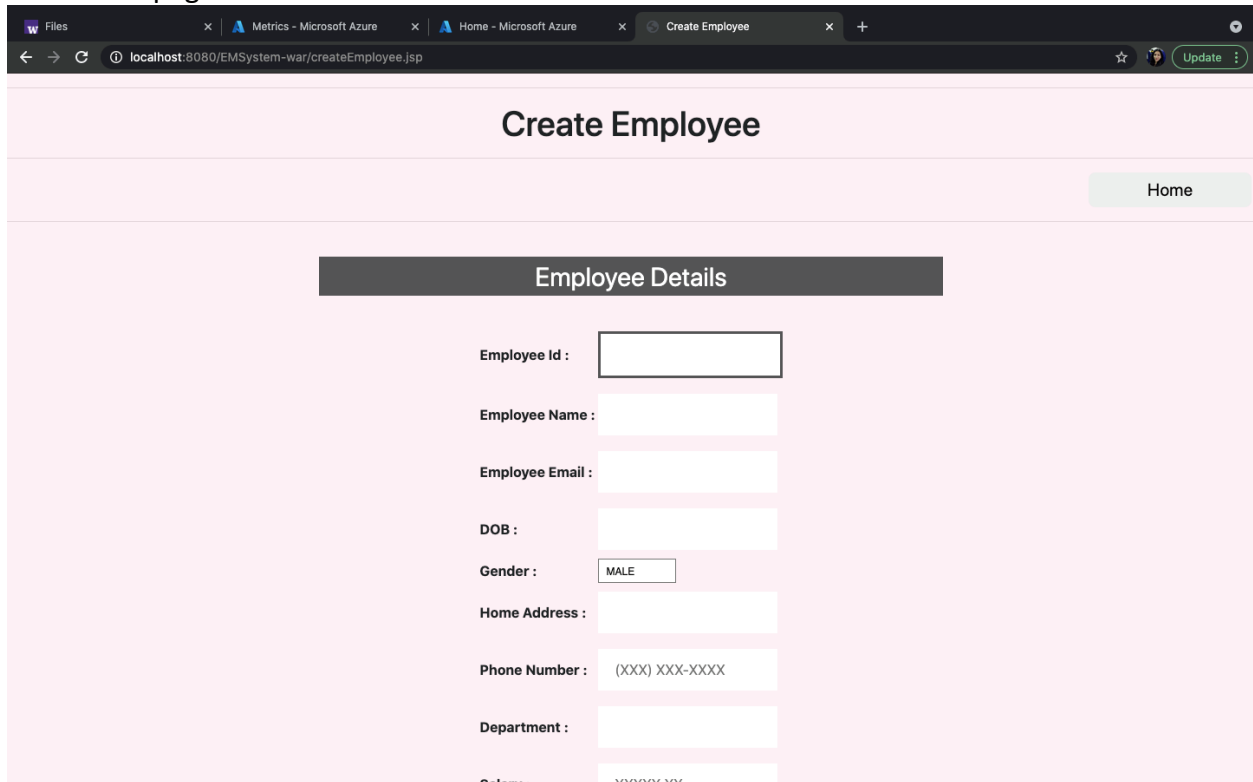
After Login is successful upon authentication, home.jsp will display the “HR Dashboard” as shown with the different scenarios and a “logout” button to logout and back to login page.



We will first create an employee with employee details



Enter the employee details to create a new employee and “Home” button to revert to dashboard page



Create Employee

[Home](#)

Employee Details

Employee Id :

Employee Name :

Employee Email :

DOB :

Gender :

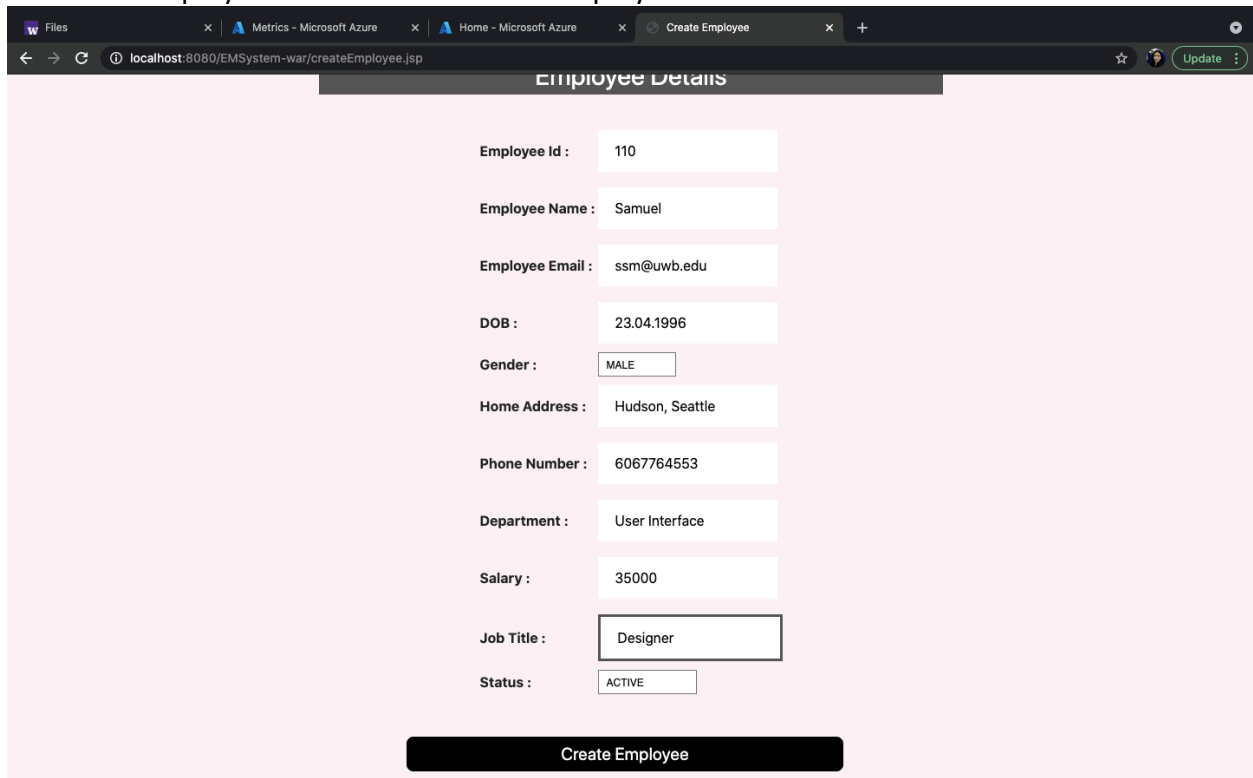
Home Address :

Phone Number :

Department :

Salary :

Create an employee name “Samuel” with Employee id “110”



Employee Details

Employee Id :

Employee Name :

Employee Email :

DOB :

Gender :

Home Address :

Phone Number :

Department :

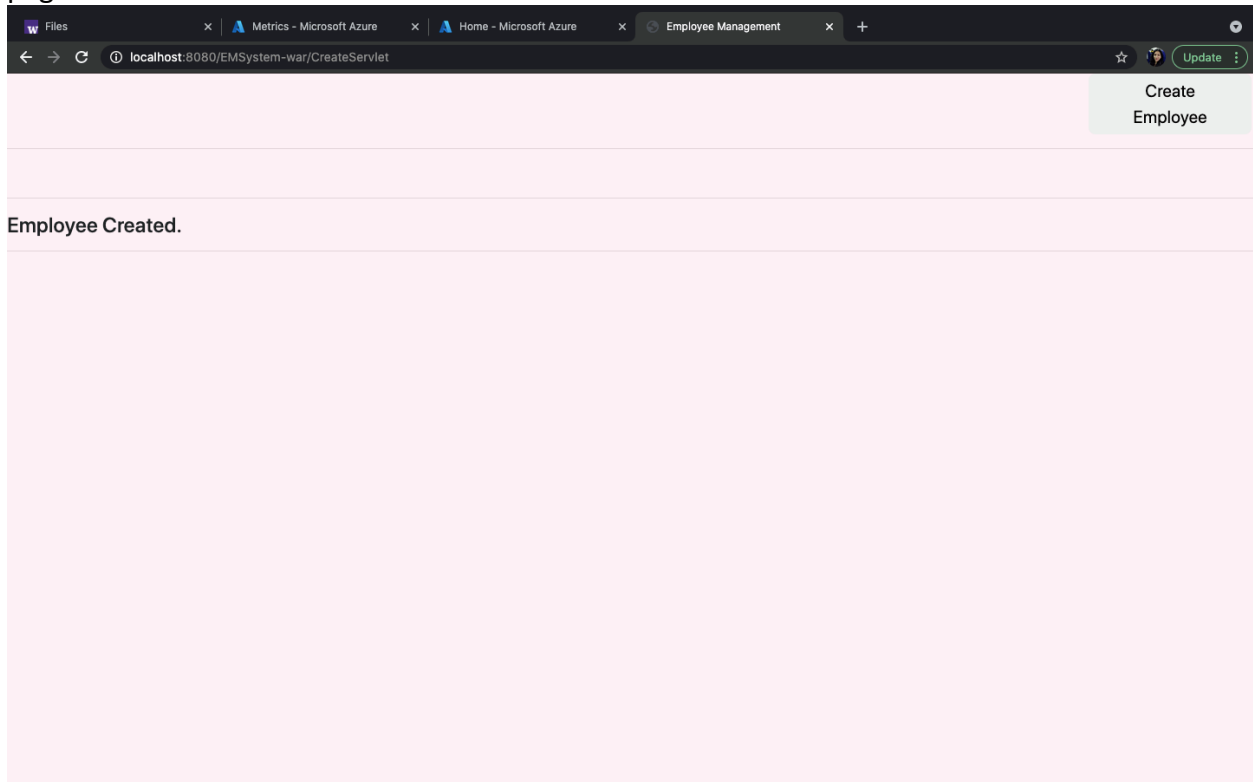
Salary :

Job Title :

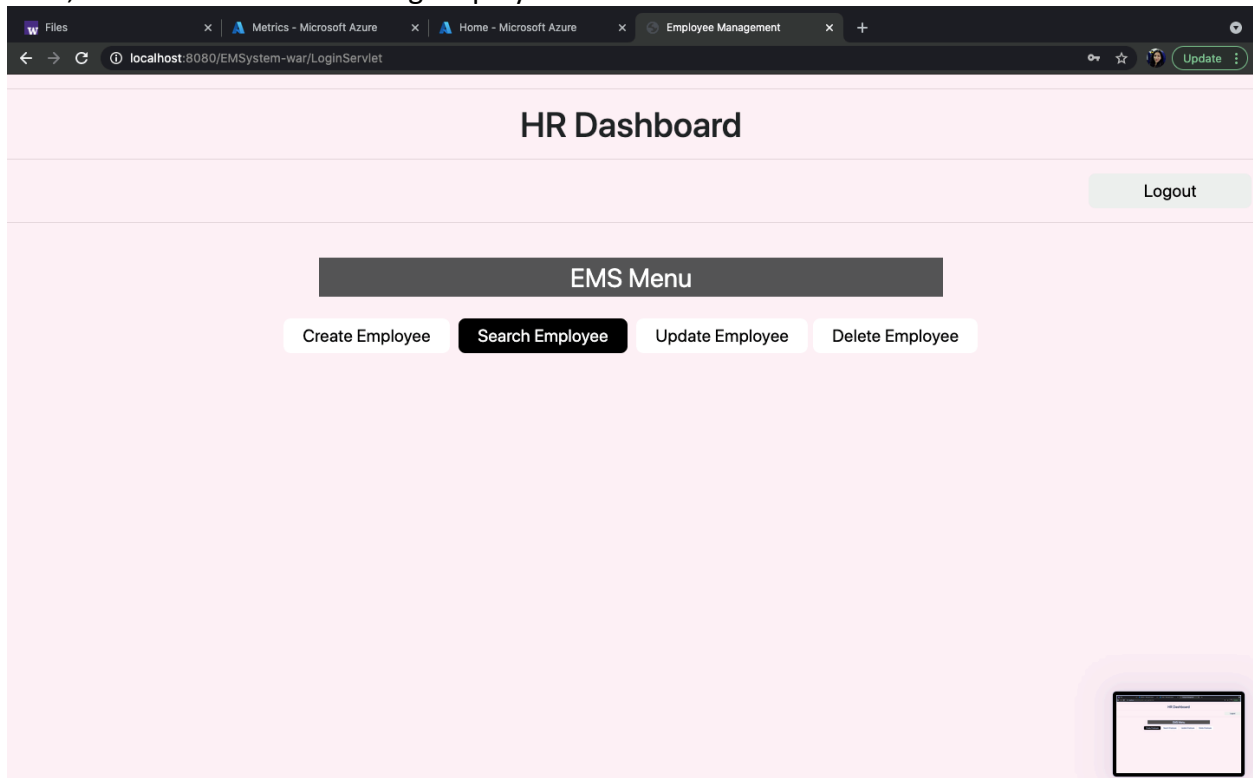
Status :

Create Employee

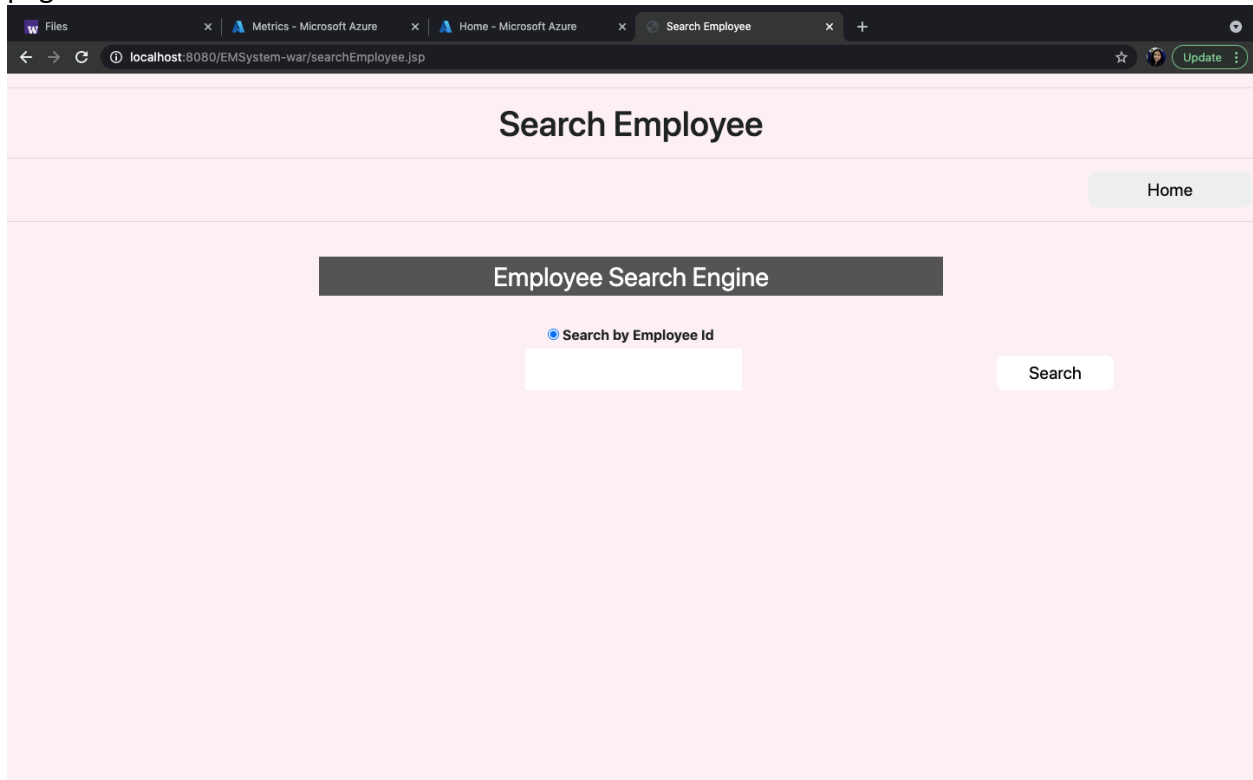
Employee created is displayed and button “Create Employee” to revert to create employee page



Now, we will search the existing employees.

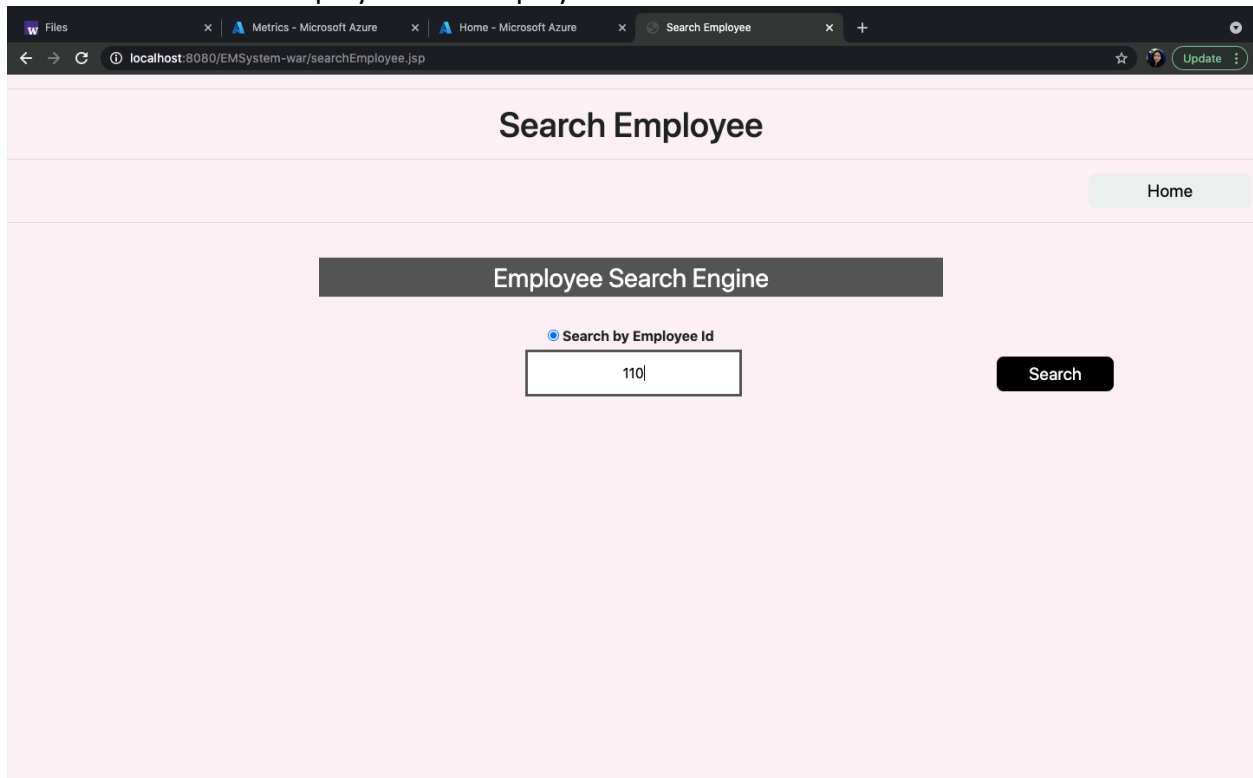


Enter the employee id to get the employee details and “Home” button to revert to dashboard page



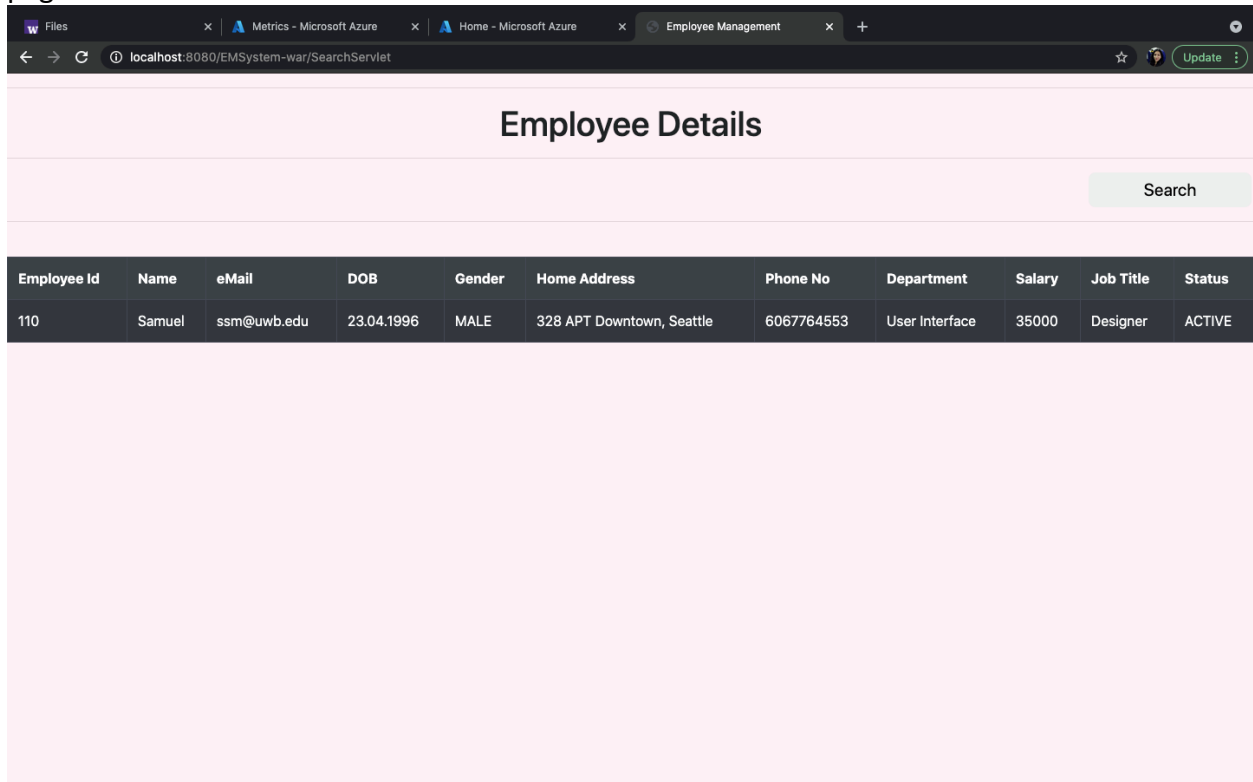
The screenshot shows a web browser window with the URL `localhost:8080/EMSystem-war/searchEmployee.jsp`. The page has a light pink background. At the top, there's a header with the title "Search Employee" and a "Home" button on the right. Below the header, there's a dark grey box with the text "Employee Search Engine". Underneath that, there's a blue link "Search by Employee Id". Below the link is a white text input field. To the right of the input field is a white button with the text "Search".

I want to search the employee with employee id “110” which I created before.



The screenshot shows the same web browser window as before, but now the text input field contains the value "110". The "Search" button is now dark grey with white text.

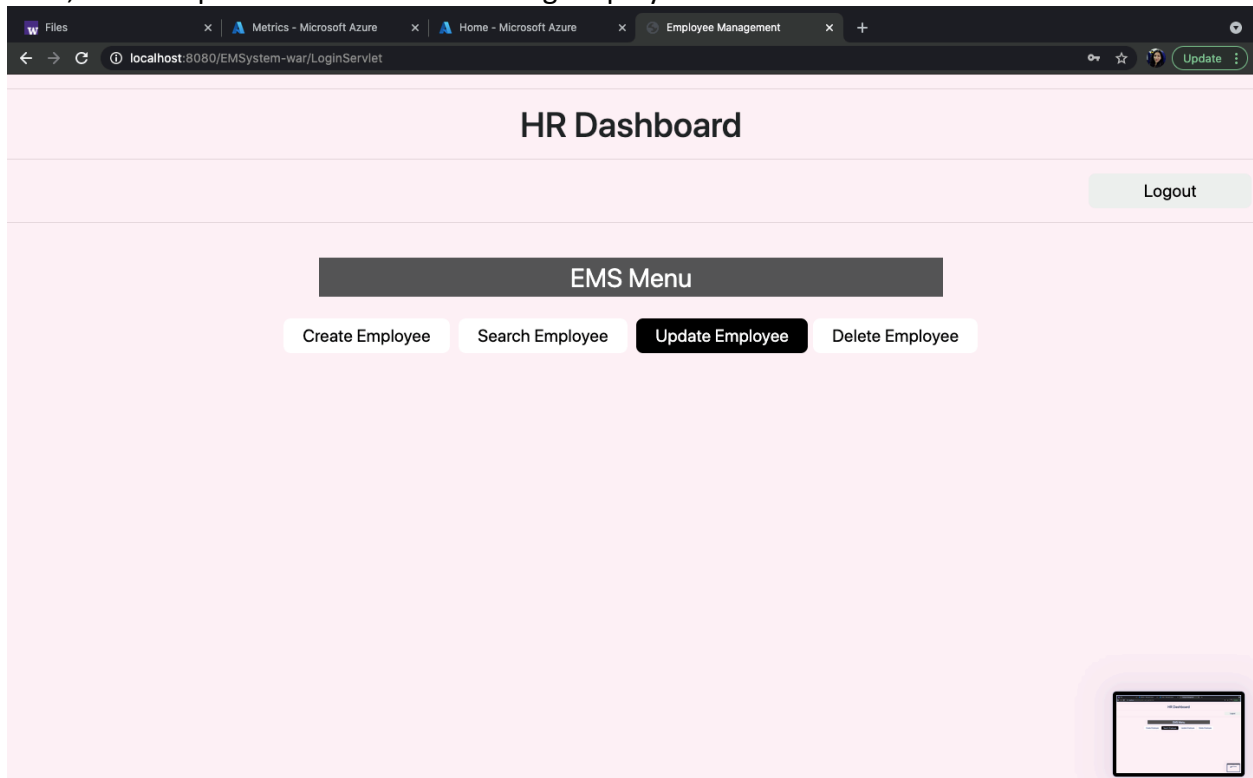
Employee details will be displayed in the next page and button “Search” to revert to search page



The screenshot shows a web browser window with the address bar displaying `localhost:8080/EMSystem-war/SearchServlet`. The page title is "Employee Details". Below the title, there is a "Search" button. A table displays employee information:

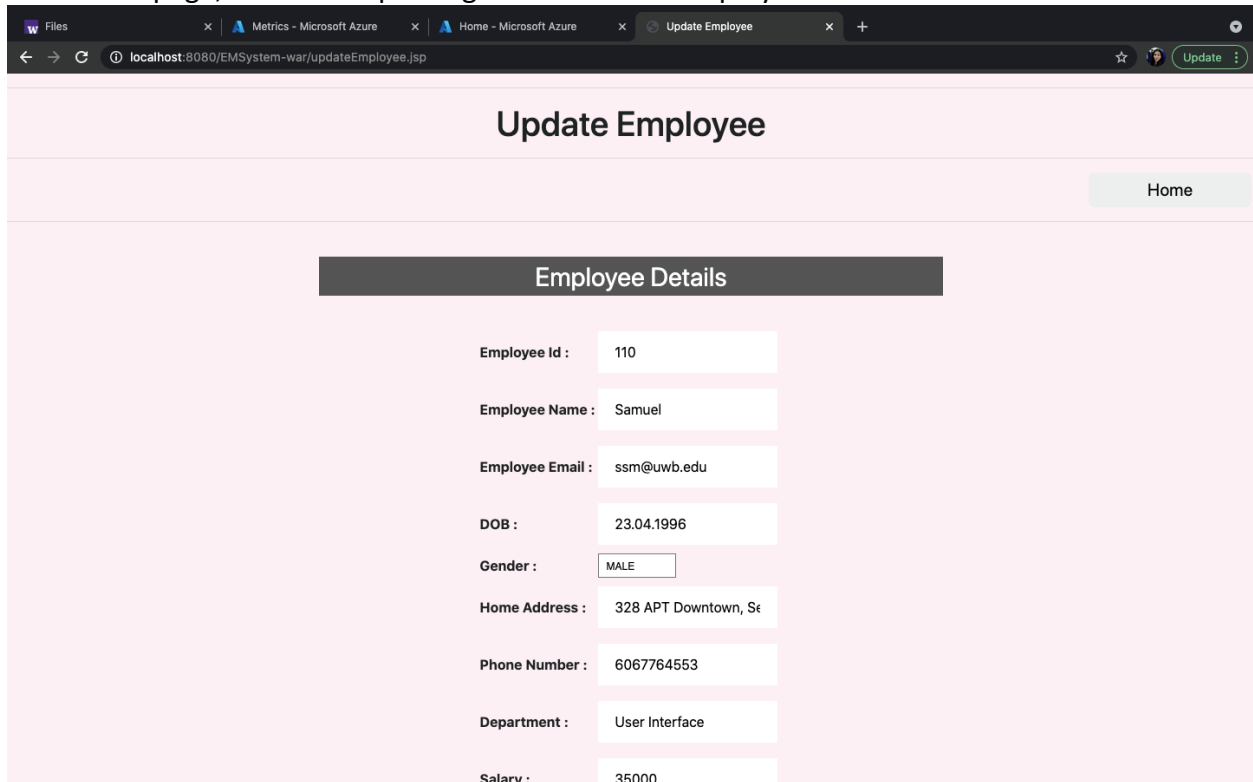
Employee Id	Name	eMail	DOB	Gender	Home Address	Phone No	Department	Salary	Job Title	Status
110	Samuel	ssm@uwb.edu	23.04.1996	MALE	328 APT Downtown, Seattle	6067764553	User Interface	35000	Designer	ACTIVE

Now, we will update the details of existing employee



The screenshot shows a web browser window with the address bar displaying `localhost:8080/EMSystem-war/LoginServlet`. The page title is "HR Dashboard". Below the title, there is a "Logout" button. In the center, there is an "EMS Menu" section with four buttons: "Create Employee", "Search Employee", "Update Employee" (highlighted in black), and "Delete Employee". A small thumbnail image of the dashboard is visible in the bottom right corner.

Enter the employee details to update the employee details and “Home” button to revert to dashboard page, here I am updating the address of employee id “110”



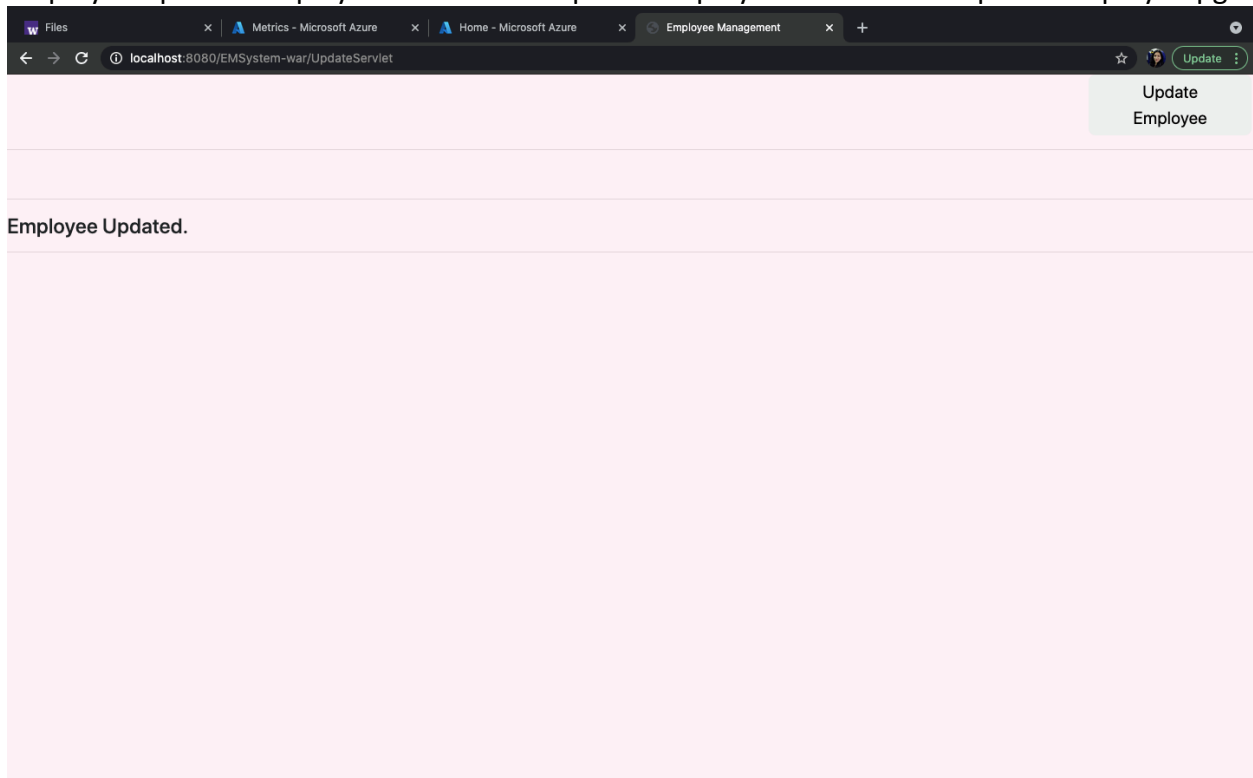
Update Employee

Home

Employee Details

Employee Id :	110
Employee Name :	Samuel
Employee Email :	ssm@uwb.edu
DOB :	23.04.1996
Gender :	MALE
Home Address :	328 APT Downtown, St
Phone Number :	6067764553
Department :	User Interface
Salary :	35000

Employee updated displayed and button “update Employee” to revert to update employee pg



Update Employee

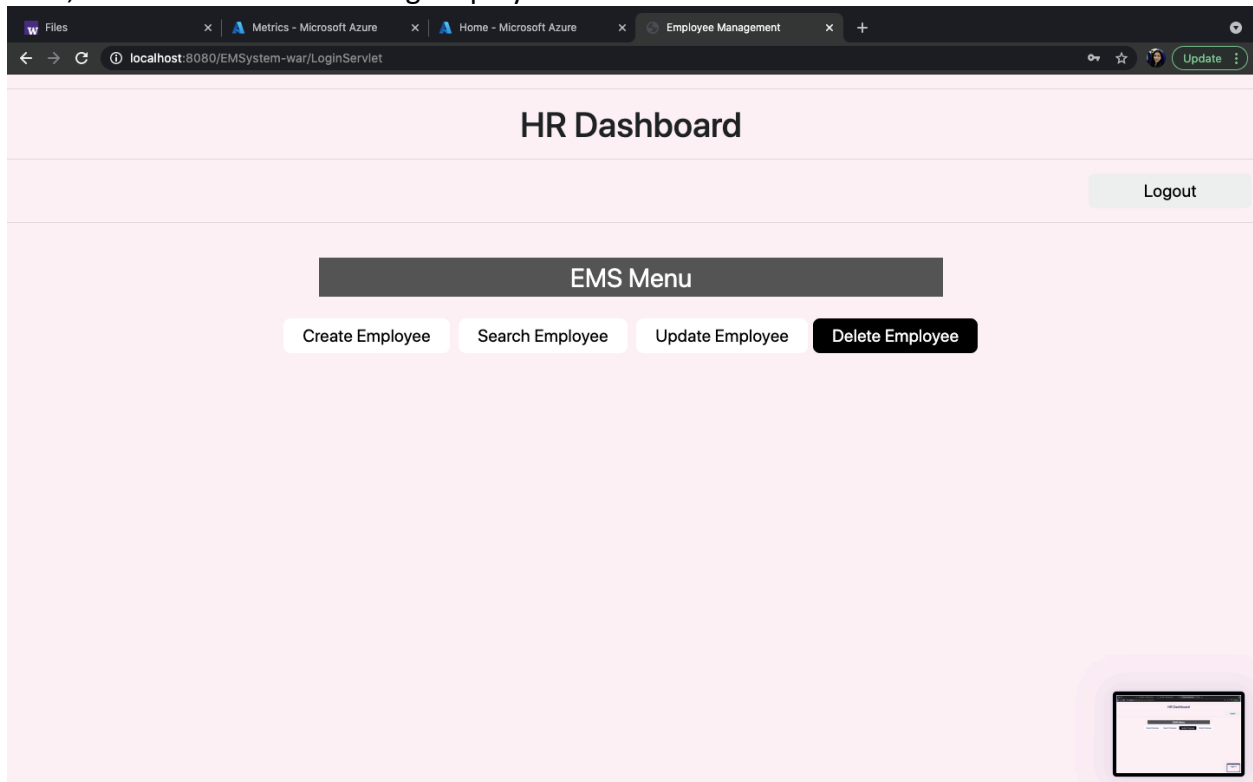
Home

Employee Details

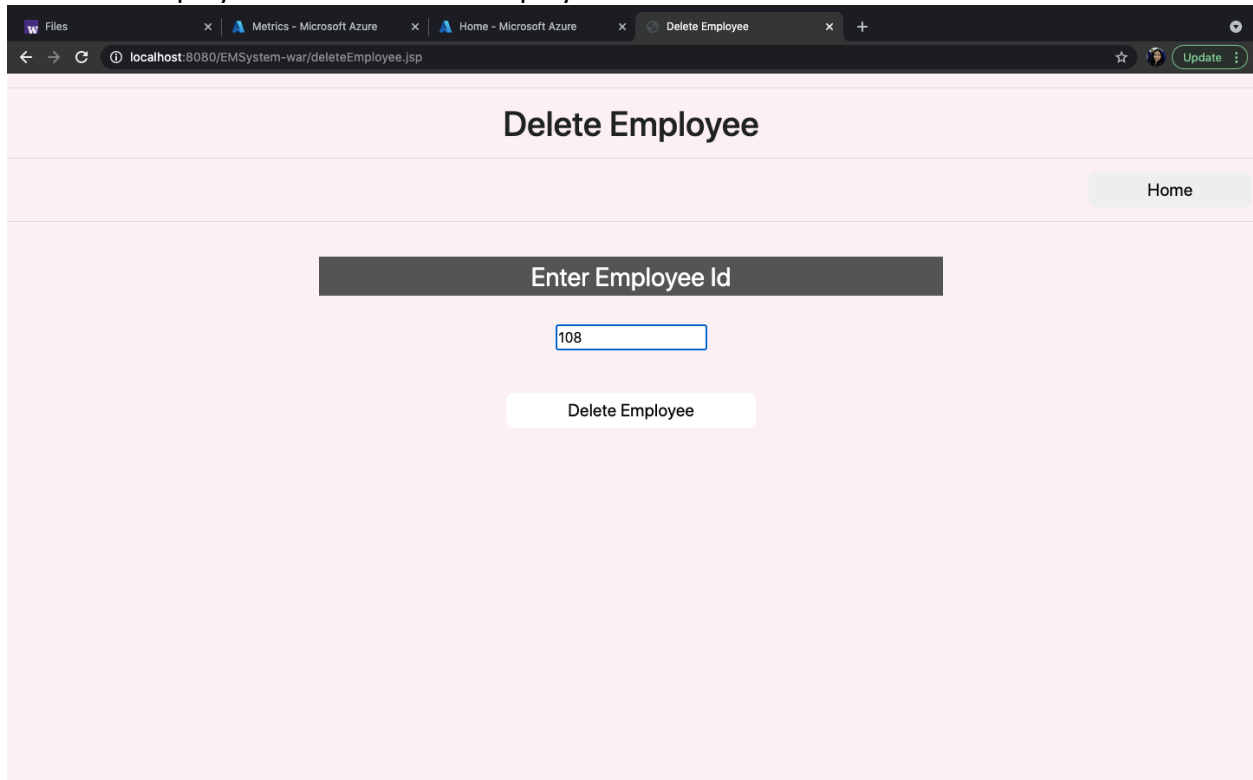
Employee Id :	110
Employee Name :	Samuel
Employee Email :	ssm@uwb.edu
DOB :	23.04.1996
Gender :	MALE
Home Address :	328 APT Downtown, St
Phone Number :	6067764553
Department :	User Interface
Salary :	35000

Employee Updated.

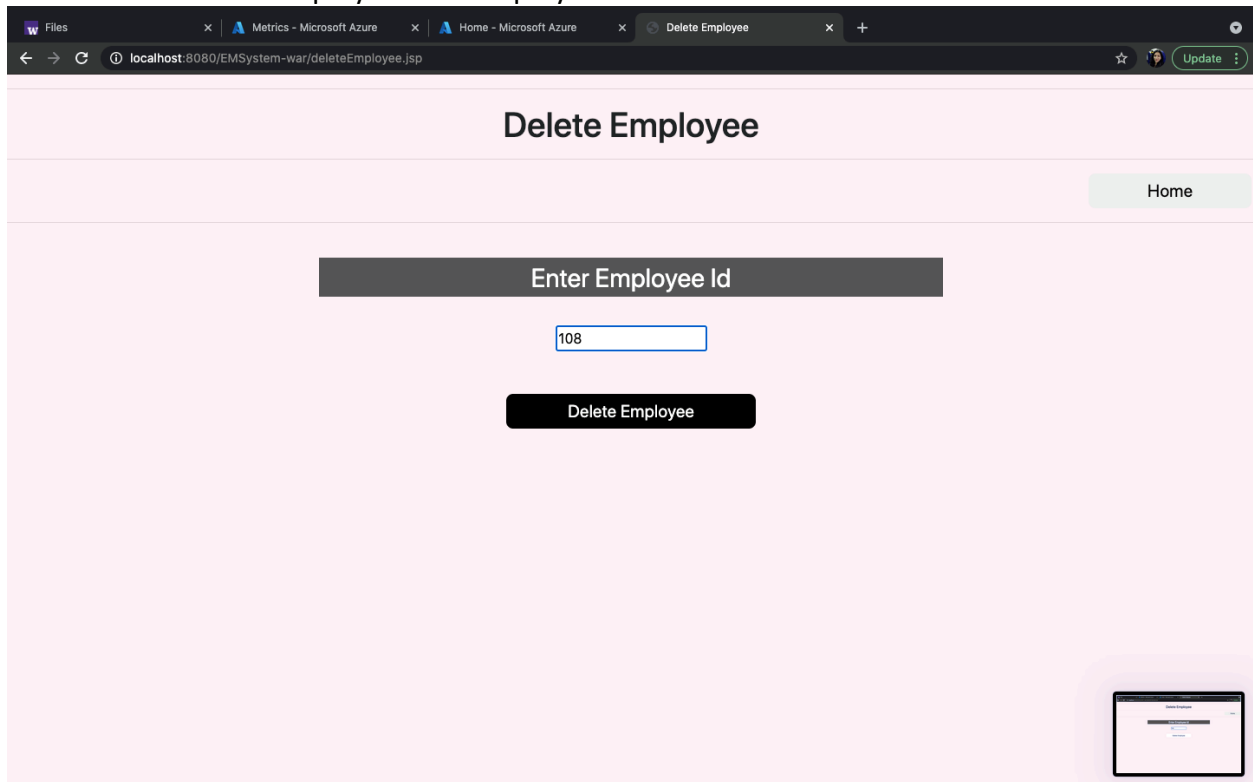
Now, we will delete the existing employee



Enter the employee id to delete the employee and "Home" button to revert to dashboard

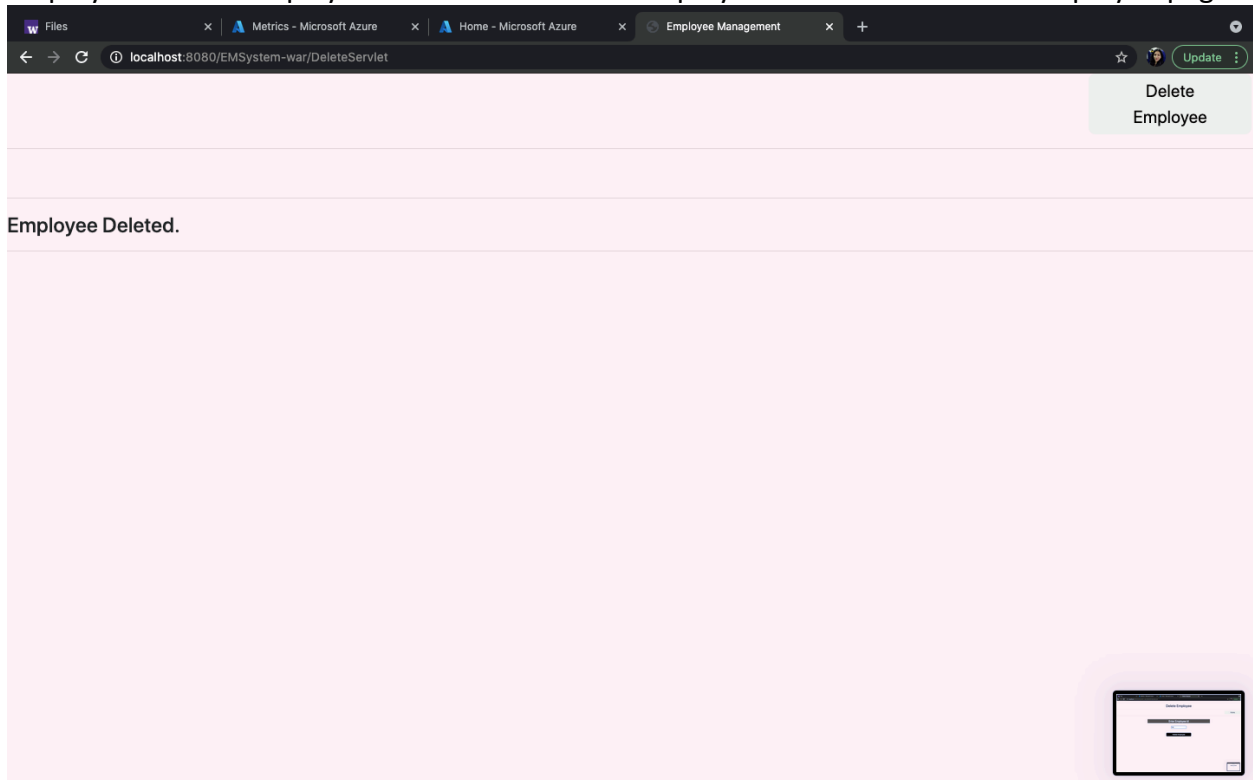


I want to delete the employee with employee id “108” which I created before.



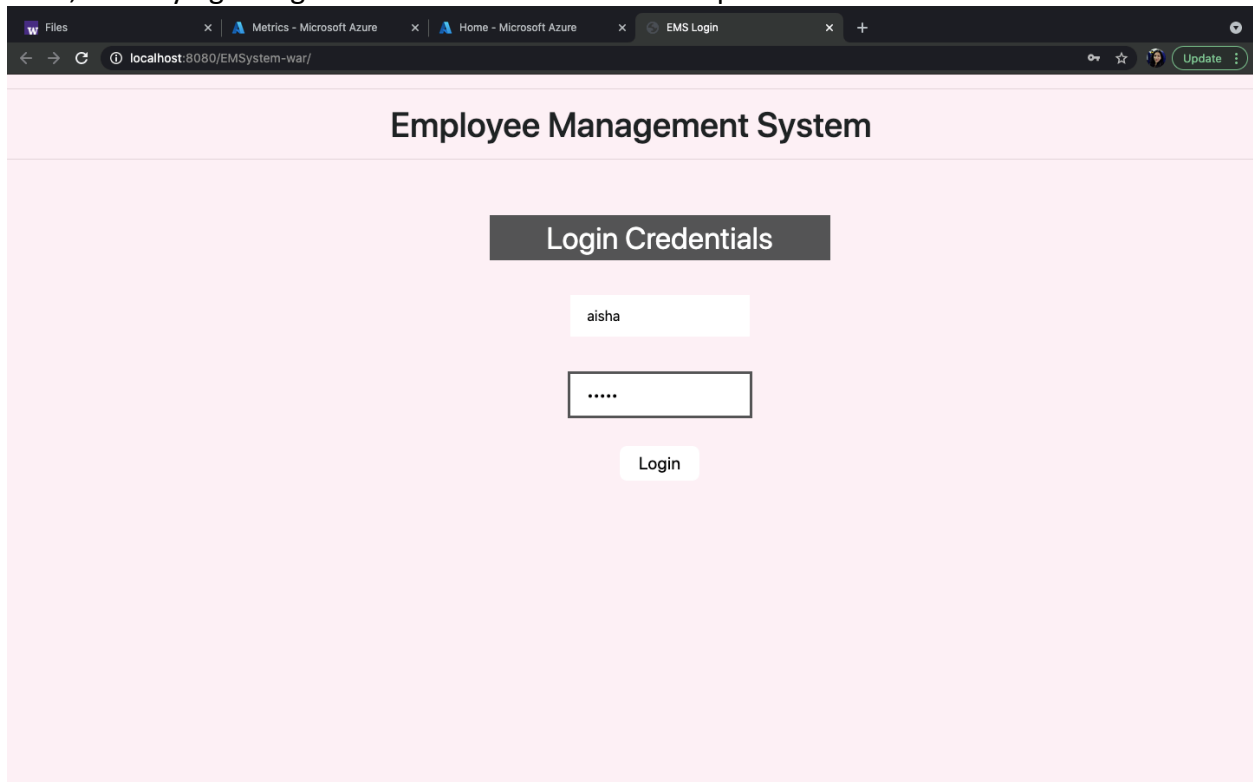
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/EMSystem-war/deleteEmployee.jsp'. The page has a light pink background and a dark header bar. The header bar contains the title 'Delete Employee' in the center and a 'Home' button on the right. Below the header, there is a dark gray rectangular box with the text 'Enter Employee Id' in white. Underneath this box is a text input field containing the number '108'. Below the input field is a dark gray button with the text 'Delete Employee' in white. In the bottom right corner of the page, there is a small thumbnail image of the application interface.

Employee deleted displayed and button “delete Employee” to revert to delete employee page



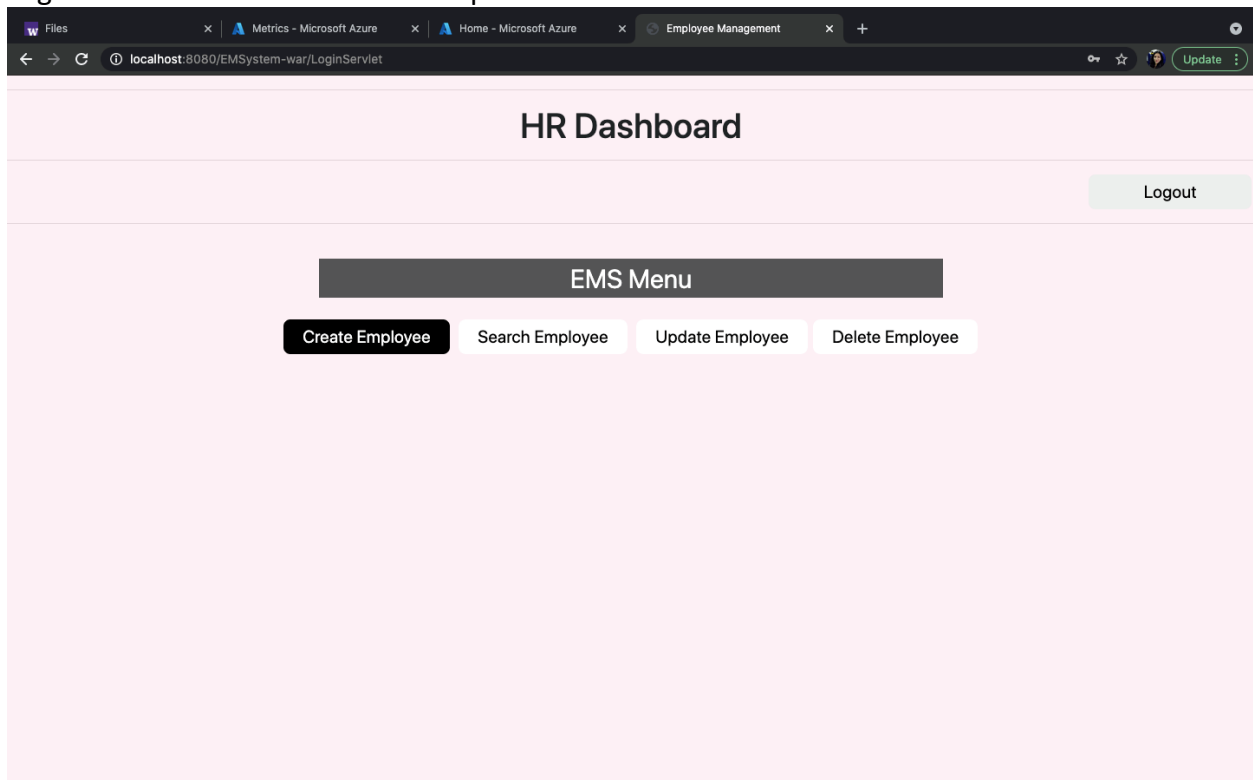
The screenshot shows the same web browser window, but the address bar now displays 'localhost:8080/EMSystem-war/DeleteServlet'. The page has a light pink background and a dark header bar. The header bar contains the title 'Delete Employee' in the center and a 'Delete Employee' button on the right. Below the header, the text 'Employee Deleted.' is displayed in a dark font. In the bottom right corner of the page, there is a small thumbnail image of the application interface.

Here, I am trying to login with different username and password “aisha”



The screenshot shows a web browser window with the URL `localhost:8080/EMSystem-war/`. The page has a light pink background and a dark header bar. The header bar contains several tabs: "Files", "Metrics - Microsoft Azure", "Home - Microsoft Azure", and "EMS Login". The main content area is titled "Employee Management System" in a large, bold, black font. Below the title, there is a dark gray box labeled "Login Credentials". Inside this box, there are two input fields: the first contains the text "aisha", and the second contains five dots, indicating a password field. Below the input fields, there is a "Login" button.

Login is successful and she can also perform the functionalities.



The screenshot shows a web browser window with the URL `localhost:8080/EMSystem-war/LoginServlet`. The page has a light pink background and a dark header bar. The header bar contains several tabs: "Files", "Metrics - Microsoft Azure", "Home - Microsoft Azure", and "Employee Management". The main content area is titled "HR Dashboard" in a large, bold, black font. In the top right corner, there is a "Logout" button. Below the title, there is a dark gray box labeled "EMS Menu". Inside this box, there are four buttons: "Create Employee", "Search Employee", "Update Employee", and "Delete Employee".

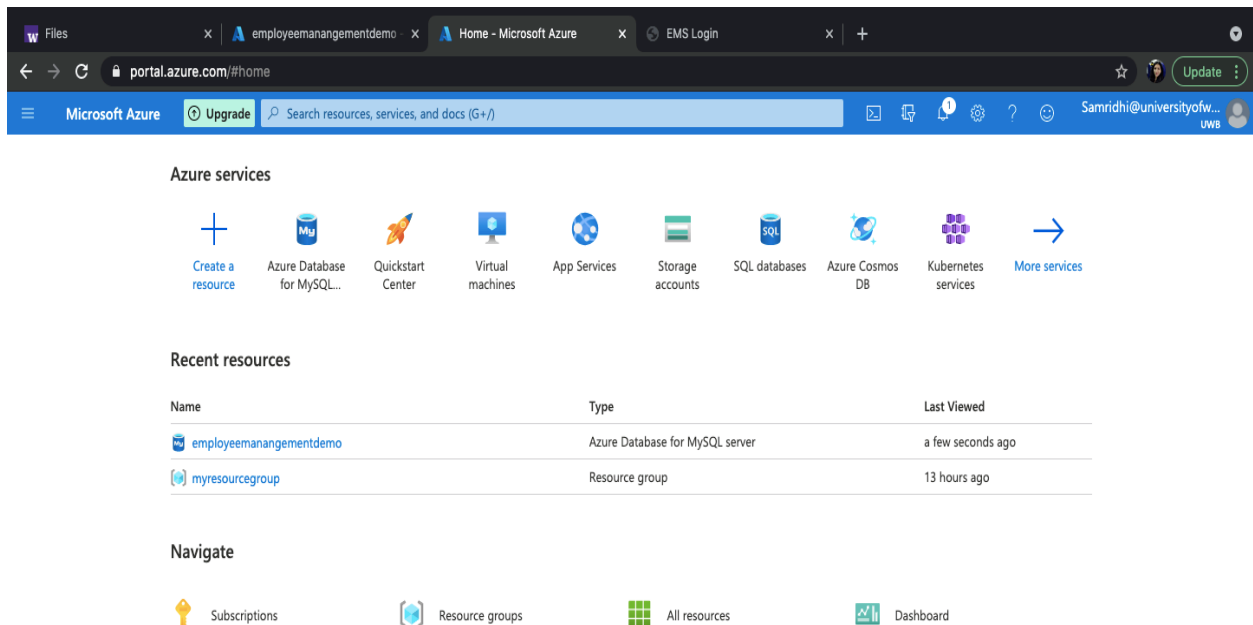
We have seen the complete flow of employee management system with user logins. I have shown two different users can login and access the application.

Test cases:

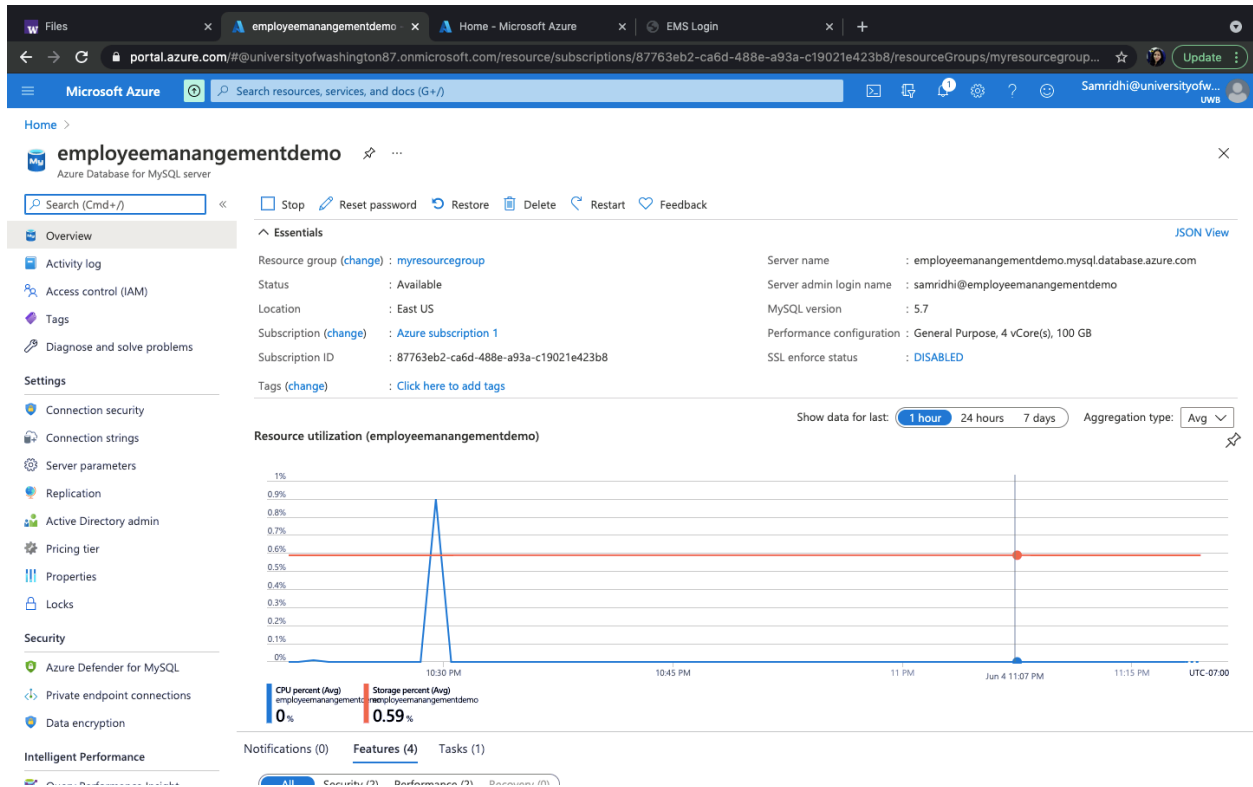
- Login into the application
- Logout from the application
- Create an employee with employee details
- Search for an employee with employee id
- Update the employee details
- Delete the employee with employee id

Snapshots of Third Tier:

Azure Portal where I have hosted my sql database



Azure Database for MySql



Azure Connection in MySQLworkbench

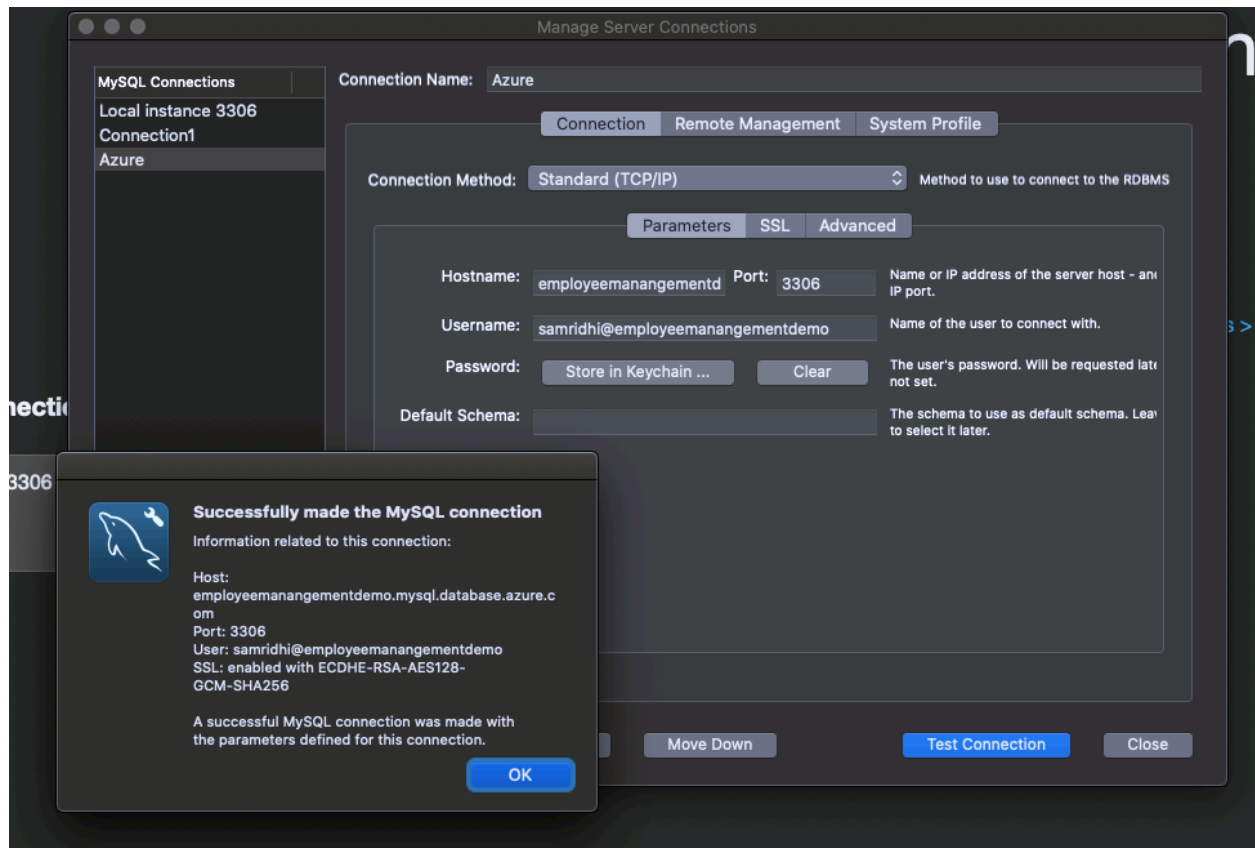
The screenshot shows the 'Manage Server Connections' dialog box in MySQL Workbench. The 'Connection Name' is set to 'Azure'. The 'Connection Method' is 'Standard (TCP/IP)'. The 'Parameters' tab is selected, showing the following configuration:

- Hostname: employeeemanagementdemo
- Port: 3306
- Username: samridhi@employeeemanagementdemo
- Password: (masked) with buttons for 'Store in Keychain ...' and 'Clear'
- Default Schema: (empty)

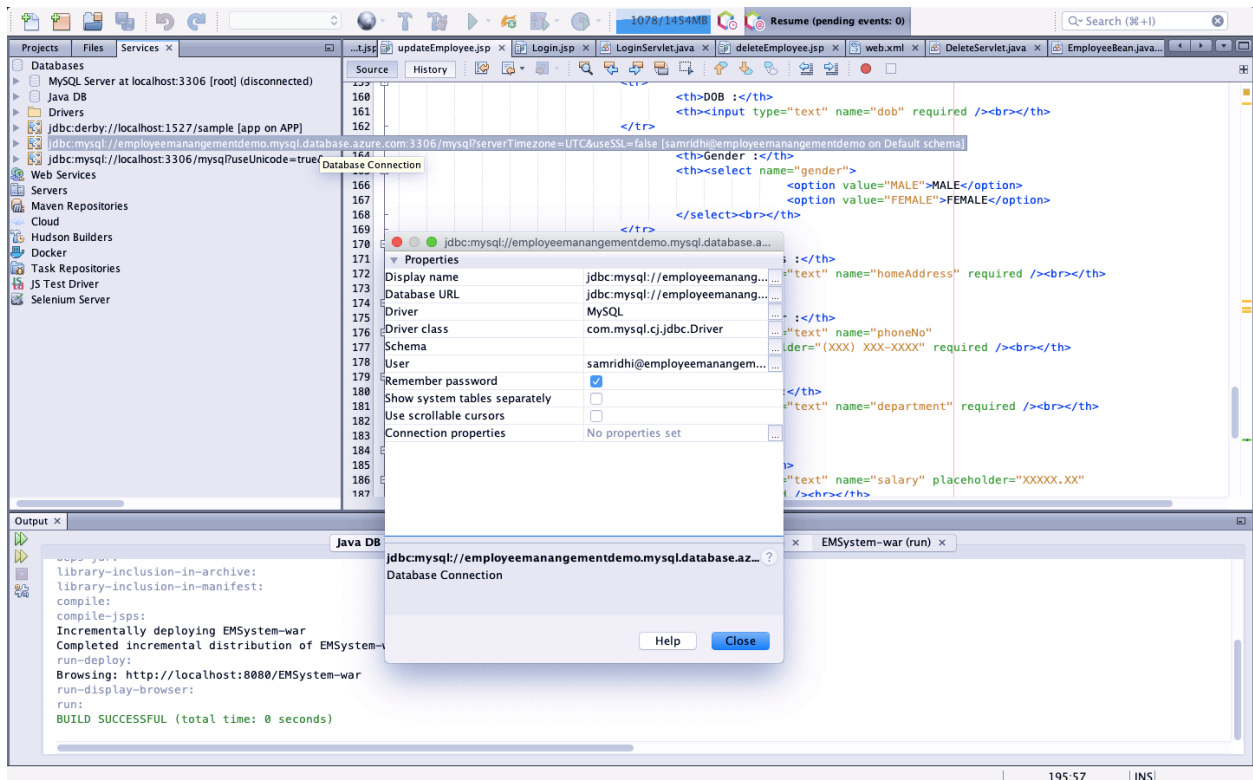
The 'System Profile' tab is also visible, showing the 'Name or IP address of the server host - an IP port.' and 'Name of the user to connect with.' fields.

At the bottom of the dialog, there are buttons for 'New', 'Delete', 'Duplicate', 'Move Up', 'Move Down', 'Test Connection', and 'Close'.

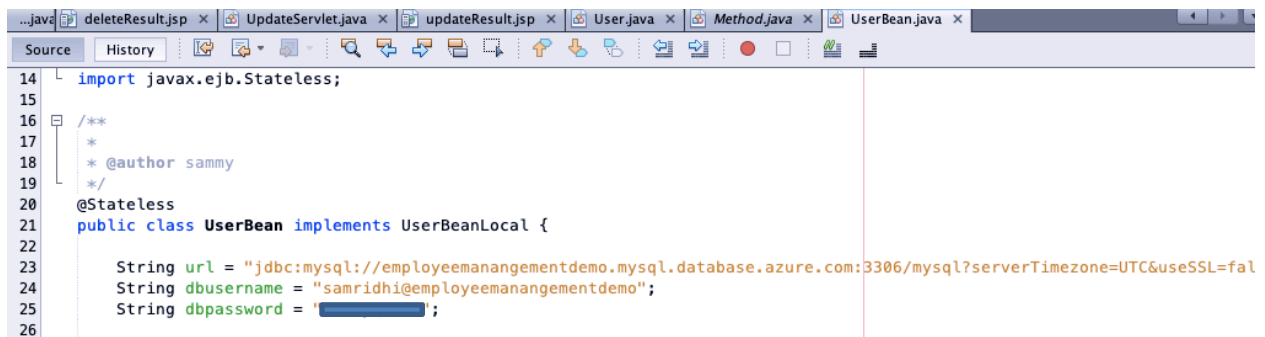
Test connection successful



Azure database connection in Netbeans editor





Usage of azure database in code



Database tables


```
1 select * from ems.login;
```

100% 25:1

Result Grid   Filter Rows:

	email	password	
▶	admin	admin	
	aisha	aisha	
	sagra15	sagra15	
	NULL	NULL	

[illegible]

Discussions:

Performance

- **High Speed and Minimal Down Time:** High availability architectures mean high speed connectivity and data retrieval as well as low down time with Azure SQL Database. This determines whether the software product responds rapidly. Currently, application is fast as it has simple use of variables, server, db.
- **Application Scalability:** It determines amount of load the software product can handle at a time. The application is highly scalable and can handle the load when deployed on a official server. The backend uses azure database which is on cloud.
- **Application Stability:** It determines whether the software product is stable in case of varying workloads. The application is stabilized in terms of workloads.
- **Application Complexity:** Applications and services complexity is low, as it is not a mosaic of components sourced from multiple places: data center, cloud, third-party, etc.
- **Application Design:** It impacts the performance directly when the design phase is not taken seriously and not followed throughout building the application.
- **Application Testing:** Acceptance testing, integration testing, smoke testing needs to be done at the user end.
- I have used **Stateless** sessions **beans** provide **faster** performance than **stateful beans**.

Reliability

For this software, typical expectations include:

- The cache stores active EJB instances in memory so that they are immediately available for client requests.
- The application performs the function that the user expected.
- It can tolerate the user making mistakes or using the software in unexpected ways.
- Its performance is good enough for the required use case, under the expected load and data volume.
- The system prevents any unauthorized access by checking the authentication and abuse.

Usability

- Using standard **SQL** provided by **MS Azure**, it's much easier to manage database systems without having to write or update a huge amount of code.
- **Availability and accessibility:** It's important to ensure the clients don't get an error trying to load the site. It is available and accessible when the server is up and running.
- **Clarity:** It is important to focus on what's important. Web pages in application is simple, consistent, familiarity and provide direct feedback.
- **Learnability:** The application is simple and provides the users to learn about the site when going to the next pages.
- **Relevancy:** The application name clearly states that it consist of relevant data for relevant user. The authentication part validates this point.

- **Test users:** Usability requires to test the users again and again to actually know who are the users. This application is able to perform it.

Limitations

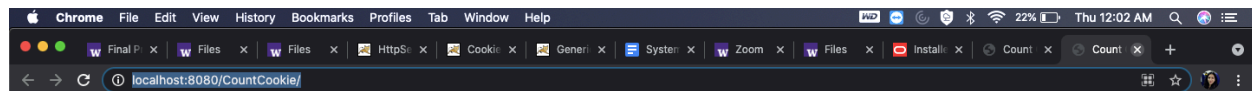
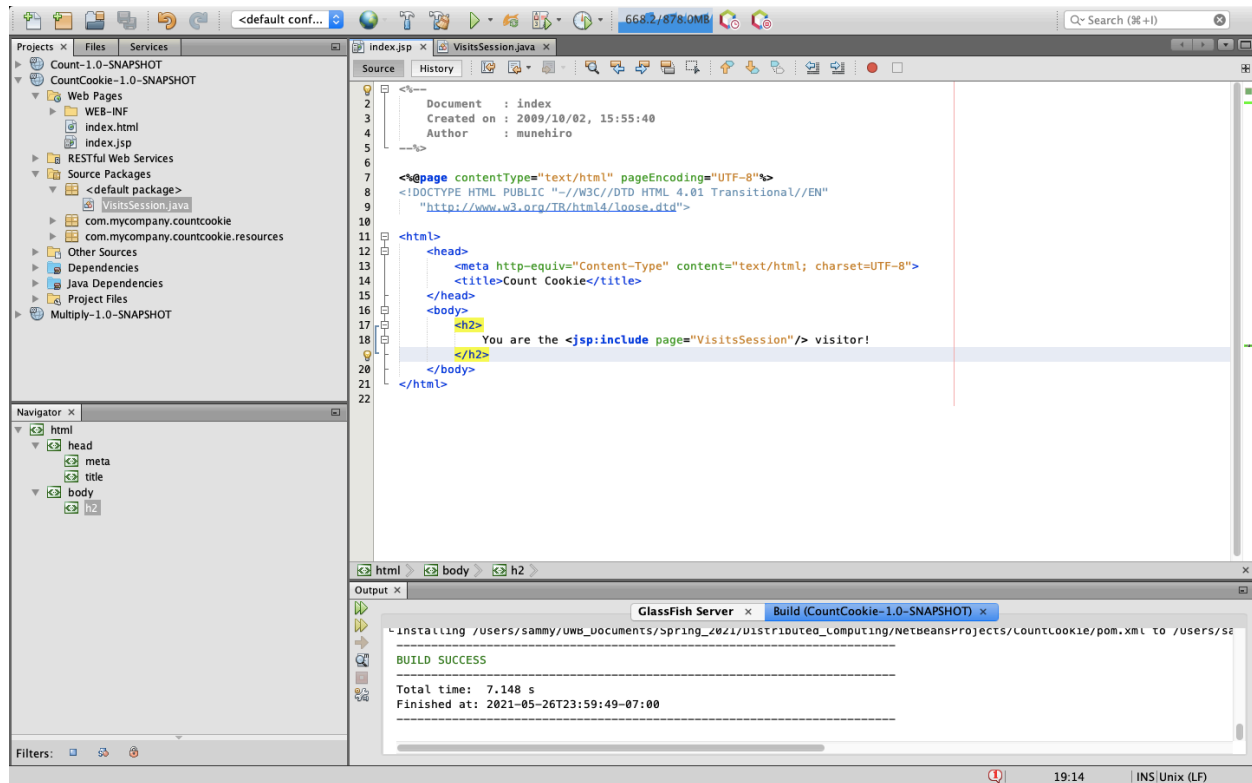
- The current application only provides the basic functionality for the human resource team i.e. add an employee, delete an employee, update employee details, search for an employee.
- The User interface is also very static, no algorithm or manipulation is running in the background, no clickstream data is cached.
- The failure-check testcases like login invalid, employee id does not exist, or invalid data type entered is not yet implemented. So, currently if any wrong detail is entered, it will lead to http error and display the exception occurred.
- Anomaly Detection in the databases such as Pager Duty to report if any of the virtual machines runs out of memory and needs to be reprocessed
- Adding photos of the employees to enable Attendance Management using Facial Recognition Technology or mapping it to the RFID attendance tracker of the employees

Possible Improvements

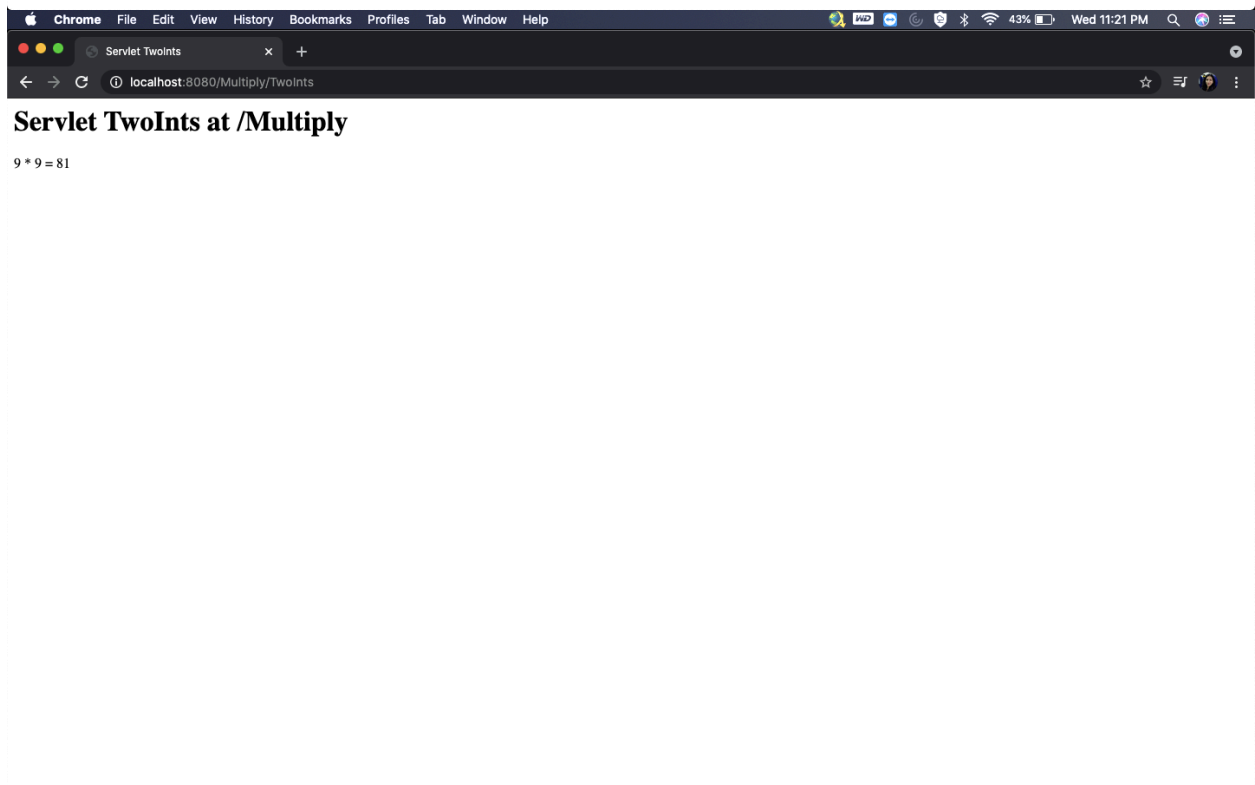
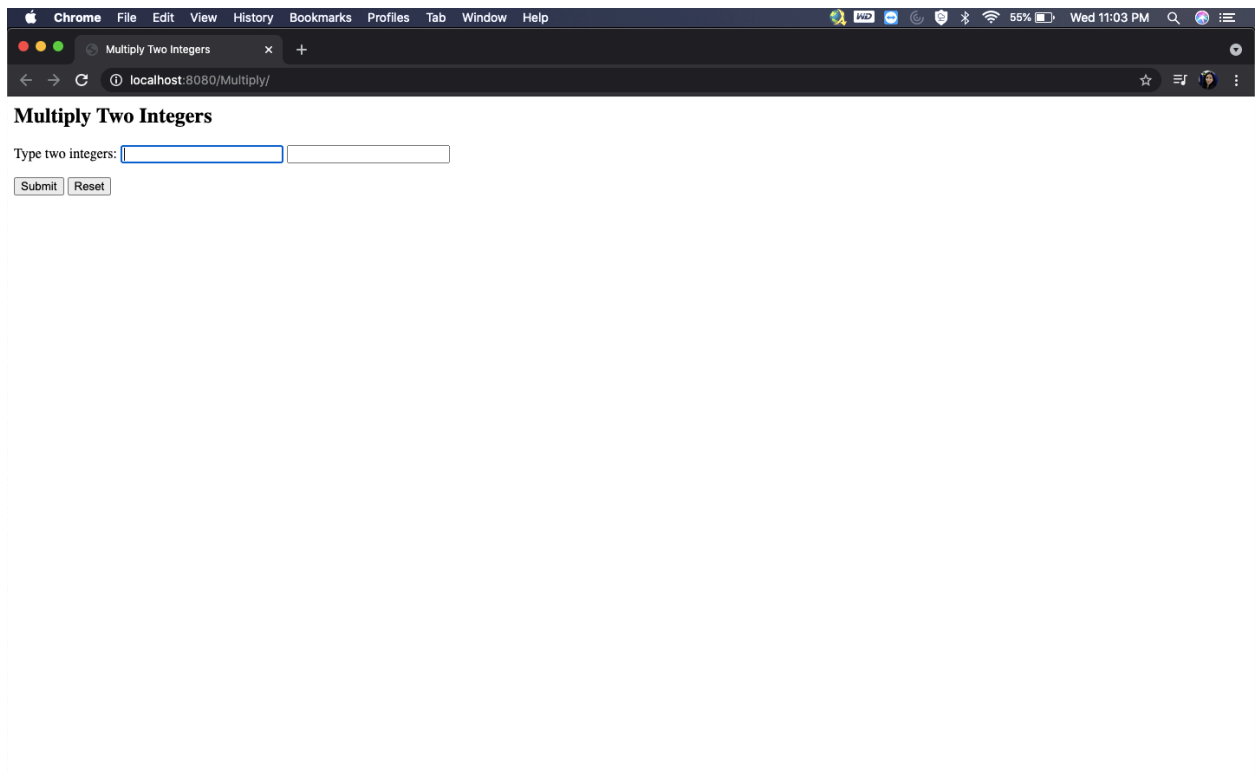
- Better usage of data structure, servers, database may decrease the execution time.
- We can improve the user interface by adding more dynamic variables like when we enter the employee id, it then hits the database and check it exists or not. If it doesn't, it should pop an alert message that please enter a valid employee id.
- We can also create an interface for the employees with limited functionality to only update its own details.
- We can add more details of the employee like their Manager's details, hierarchy in the company, expenses, leaves.
- There can be many enhancements done on the frontend part using the angular JS on the CSS side.
- We can add an announcements portal where HR can maintain its announcements shared throughout the company.
- We can also create more functionalities for the HR to view all the employees, post a comment for the employee, set a reminder to update details to the employee.
- We can also add a third-party authenticator to our application which stores the mobile number or email and send a one-time password after login page to authenticate the valid user.
- Data Security and Privacy such as KMS Key for encrypting employee data in transit and at rest, and following certification standards such as GDPR
- Tracking metrics on data ingestion using cloud services such as Cloud Watch for emitting metrics such as number of employees directly to a HR dashboard
- Easy way to upload bulk data from existing systems of records such as ERP Systems, CSV Files, using pre-configured connectors from service such as Amazon AppFlow

Lab Session 8:

Output:

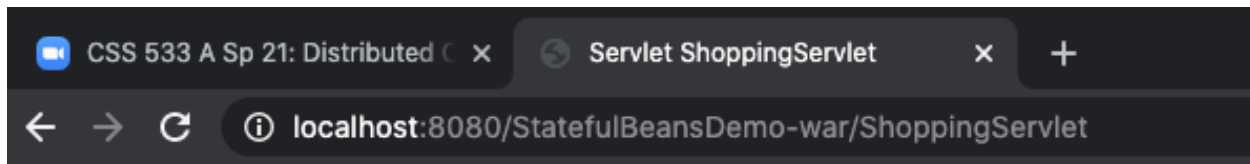
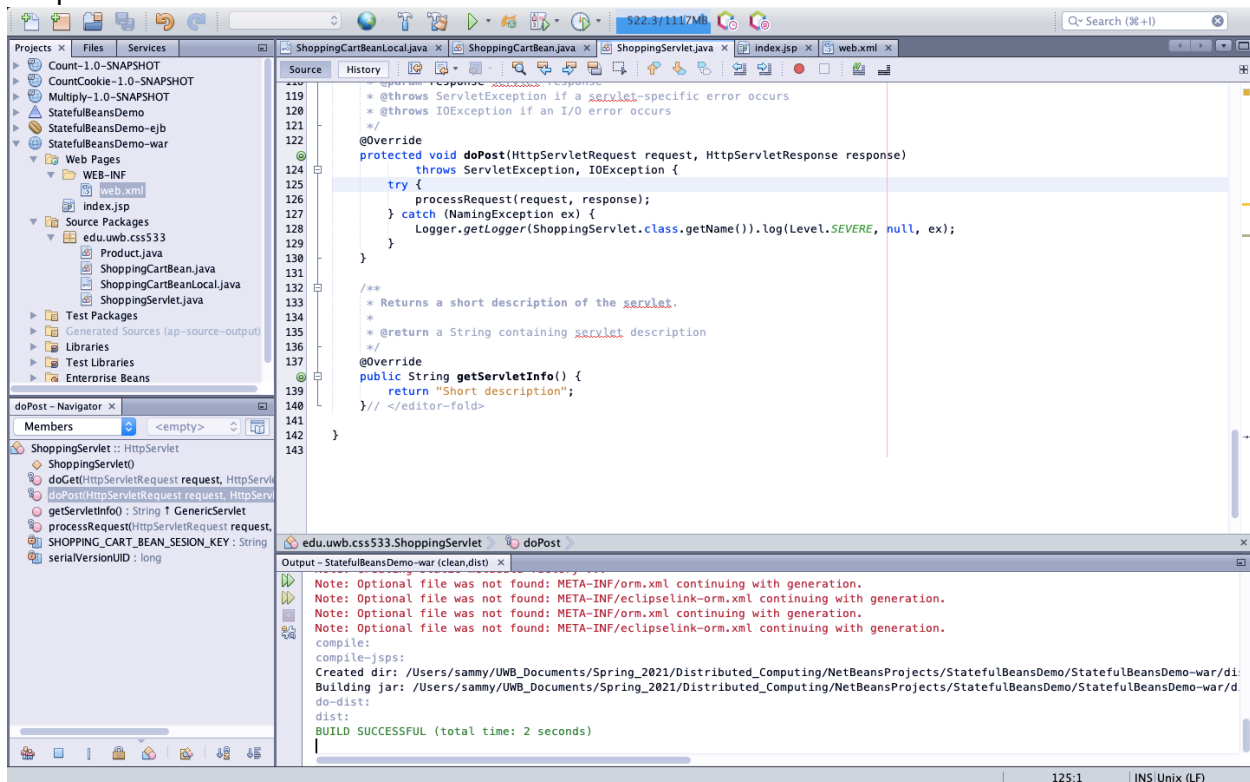


You are the 1st visitor!



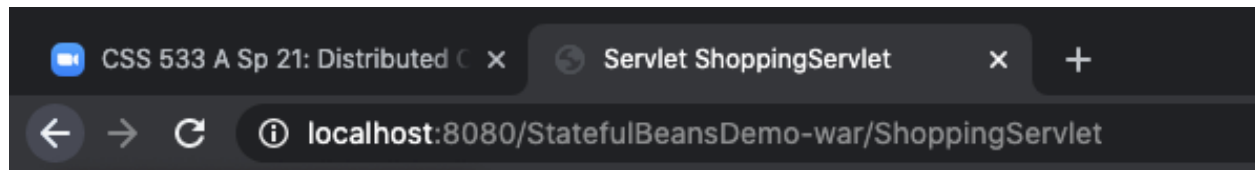
Lab Session 9:

Output:



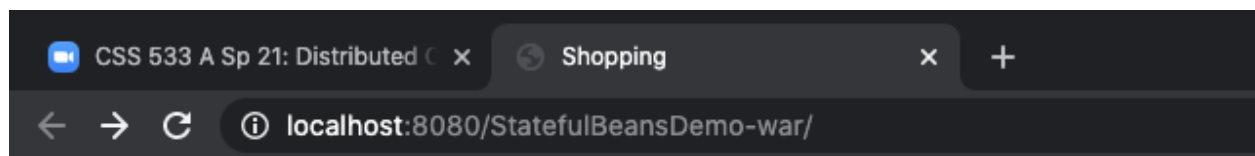
Servlet ShoppingServlet

Banana in your shopping bag



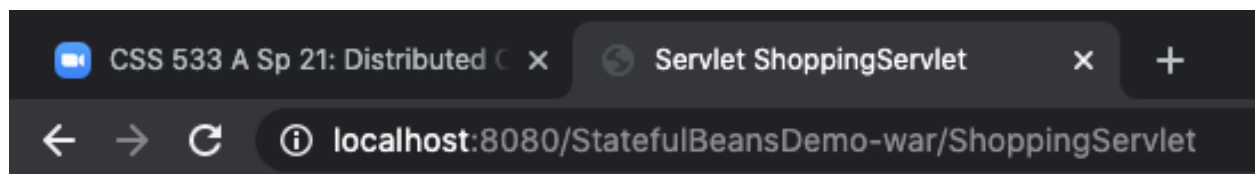
Servlet ShoppingServlet

Strawberry in your shopping bag



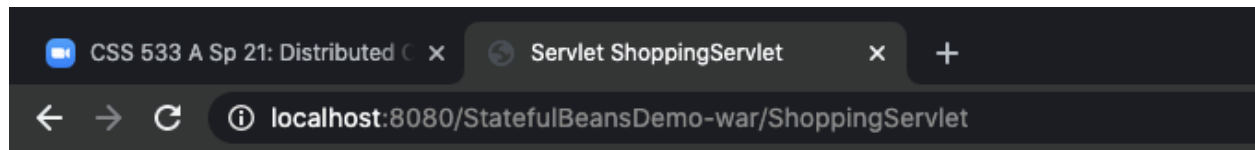
Shopping

Enter Product: Complete Shopping:



Servlet ShoppingServlet

Mango in your shopping bag



Servlet ShoppingServlet

Papaya in your shopping bag

Your items:

Apple

Banana

Mango

Strawberry

Papaya

shopping completed.....Thank you!