Ex No 2 PROJECT IMPLEMENTATION USING MVC AND SPRING FRAMEWORK

AIM:

To implement projects using MVC and Spring Framework.

INTRODUCTION:

The Model-View-Controller (MVC) architecture is a popular software design pattern used for developing user interfaces by dividing the application into three interconnected components. This separation helps in managing complex applications by organizing the codebase effectively. The Model is responsible for handling the data and the business logic of the application. The View displays the data to the user and presents the user interface, while the Controller processes user inputs, communicates with the model, and updates the view accordingly. By separating concerns, MVC enhances code reusability, scalability, and maintainability, making it easier to test and manage each component independently. This architecture is widely adopted in modern web and desktop application frameworks due to its clear structure and flexibility.

PROCEDURE:

1.Set Up the Spring Boot Project

- Create a new Spring Boot project using Spring Initializr or your IDE.
- Add the necessary dependencies: Spring Web, Spring Data JPA, Thymeleaf, MySQL Driver.
- Create the base package structure:
  - controller – for handling HTTP requests.
  - domain – for entity classes.
  - repository – for data access logic.
  - service – for implementing business logic.

2.Design the Model Layer (Domain Classes)

- Create a class Login.java annotated with @Entity to store user credentials.
- Include fields: username and password.
- Provide constructors, getters, and setters.
- Create a class Order.java annotated with @Entity to store cafe order data.
- Include fields: id, customerName, email, drinkName, size, quantity, date.
- Use @GeneratedValue for automatic ID generation.

3.Create the Repository Interfaces

- Define RegRepo.java extending JpaRepository<Login, String> to register users.

    - Include the method findByUsername(String username).
- Define LoginRepo.java extending JpaRepository<Login, String> to validate login.
    - Include the method findByUsernameAndPassword(String username, String password).
- Define OrderRepository.java extending JpaRepository<Order, Long> to save order records.

4.Implement the Service Layer

- Create RegService.java:
    - Inject RegRepo using @Autowired.
    - Implement a method to check if a username exists and save a new user.
- Create LoginService.java:
    - Inject LoginRepo using @Autowired.
    - Implement a method to check credentials and return users.
- Create OrderService.java:
    - Inject OrderRepository using @Autowired.
    - Implement a method to save customer orders.

5.Develop the Controller Layer:

- Create RegController.java:
    - Handle GET /register to show the registration page.
    - Handle POST /register to save the user if not already present.
- Create LoginController.java:
    - Handle GET / to show the login page.
    - Handle POST /login to validate login and redirect accordingly.
- Create HomeController.java:
    - Handle GET /home to show the main cafe services page after login.
- Create OrderController.java:
    - Handle GET /order to show the order placement form.
    - Handle POST /order to save order and redirect to confirmation.
    - Handle GET /order-confirmation to show order details.

6.Create Thymeleaf HTML Templates in src/main/resources/templates

- register.html – Registration form.
- login.html – Login form.
- home.html – Welcome or dashboard page.
- order-form.html – Order placement form.
- order-confirmation.html – Displays order summary.

7.Connect to the Database

- Configure application.properties to connect to the MySQL database.
- Enable JPA and auto table creation using Hibernate.

8.Run and Test the Application

- Start the Spring Boot application.
- Open the browser and:
    - Register a new user via /register.
    - Log in with correct credentials via /login.
    - Place an order using the order form at /order.
    - View order confirmation details at /order-confirmation.

CODE:

LoginController.java:

```
package com.batch2.artifact1.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestParam;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.service.LoginService;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class LoginController {

    @Autowired
    private LoginService service;

    // Show login page on root or "/login"
    @GetMapping({"/", "/login"})
    public String showLoginPage(@RequestParam(required = false) String error, Model model) {
        if (error != null) {
            model.addAttribute("error", "Invalid username or password");
        }
        return "login"; // login.html
    }

    // Home page after login
    @GetMapping("/home")
    public String homePage() {
        return "home";
    // home.html (Lattefy dashboard)
    }

    // Your Orders page
    @GetMapping("/urorder")
    public String orderPage() {
        return "urorderrec"; // urorderrec.html
    }

    // Favorites page
    @GetMapping("/favorites")
    public String favoritesPage() {
        return "favorites"; // favorites.html
    }

    // Drink Menu (default landing on Coffee tab)
    @GetMapping("/drink-menu-coffee")
    public String drinkMenuCoffeePage() {
        return "drink-menu-coffee"; // drink-menu-coffee.html
    }

    // Drink Menu - Chocolate category
    @GetMapping("/drink-menu-choco")
    public String drinkMenuChocoPage() {
        return "drink-menu-choco"; // drink-menu-choco.html
    }

    // Drink Menu - Others category
    @GetMapping("/drink-menu-others")
    public String drinkMenuOthersPage() {
        return "drink-menu-others"; // drink-menu-others.html
    }

    @GetMapping("/urpastorder")
```

```
    public String urPastOrderPage() {
        return "urpastorder";
    }

    // Handle login form POST
    @PostMapping("/login")
    public String processLogin(@RequestParam String username,
                               @RequestParam String password,
                               Model model) {

        Login user = service.log(username, password);

        if (user != null) {
            return "redirect:/home";
        } else {
            model.addAttribute("error", "Invalid username or password");
            return "login"; // login.html with error
        }
    }
}
```

RegController.java:

```
package com.batch2.artifact1.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.batch2.artifact1.service.RegService;

@Controller
public class RegController {
    @Autowired
    private RegService service;

    @GetMapping("/register")
    public String showRegisterPage() {
        return "register";
    }

    @PostMapping("/register")
    public String registerUser(@RequestParam String username, @RequestParam String
password, Model model) {
        boolean isRegistered = service.registerUser(username, password);

        if (isRegistered) {
            model.addAttribute("message", "Registration successful! Please login.");
            return "login";
        } else {
            model.addAttribute("error", "Username already exists!");
            return "register";
        }
    }
}
```

Login.java:

```
package com.batch2.artifact1.domain;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "login")
public class Login {
    @Id
    public String username;
    public String password;

    public Login() {
    }

    public Login(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

LoginRepo.java:

```
package com.batch2.artifact1.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.batch2.artifact1.domain.Login;

@Repository
public interface LoginRepo extends JpaRepository<Login, String> {
    Login findByUsernameAndPassword(String username, String password);
}
```

RegRepo.java:

```
package com.batch2.artifact1.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.batch2.artifact1.domain.Login;

@Repository
```

```
public interface RegRepo extends JpaRepository<Login, String> {
   Login findByUsername(String username);
}
```

LoginService.java:

```
package com.batch2.artifact1.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.repository.LoginRepo;

@Service
public class LoginService {

   @Autowired
   private LoginRepo rep;

   public Login log(String username, String password) {
      Login user = rep.findByUsernameAndPassword(username, password);
      return user;
   }
}
```

RegService.java:

```
package com.batch2.artifact1.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.repository.RegRepo;

@Service
public class RegService {
    @Autowired
    private RegRepo rep;

    public boolean registerUser(String username, String password) {

        if (rep.findByUsername(username) != null) {
            return false;
        }

        Login newUser = new Login(username, password);
        rep.save(newUser);
        return true;
    }

}
```

ArtifactApplication.java:

```
package com.batch2.artifact1.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.repository.RegRepo;

@Service
public class RegService {
    @Autowired
    private RegRepo rep;

    public boolean registerUser(String username, String password) {

        if (rep.findByUsername(username) != null) {
            return false;
        }

        Login newUser = new Login(username, password);
        rep.save(newUser);
        return true;
    }

}
```

Atrifact1ApplicationTests.java:

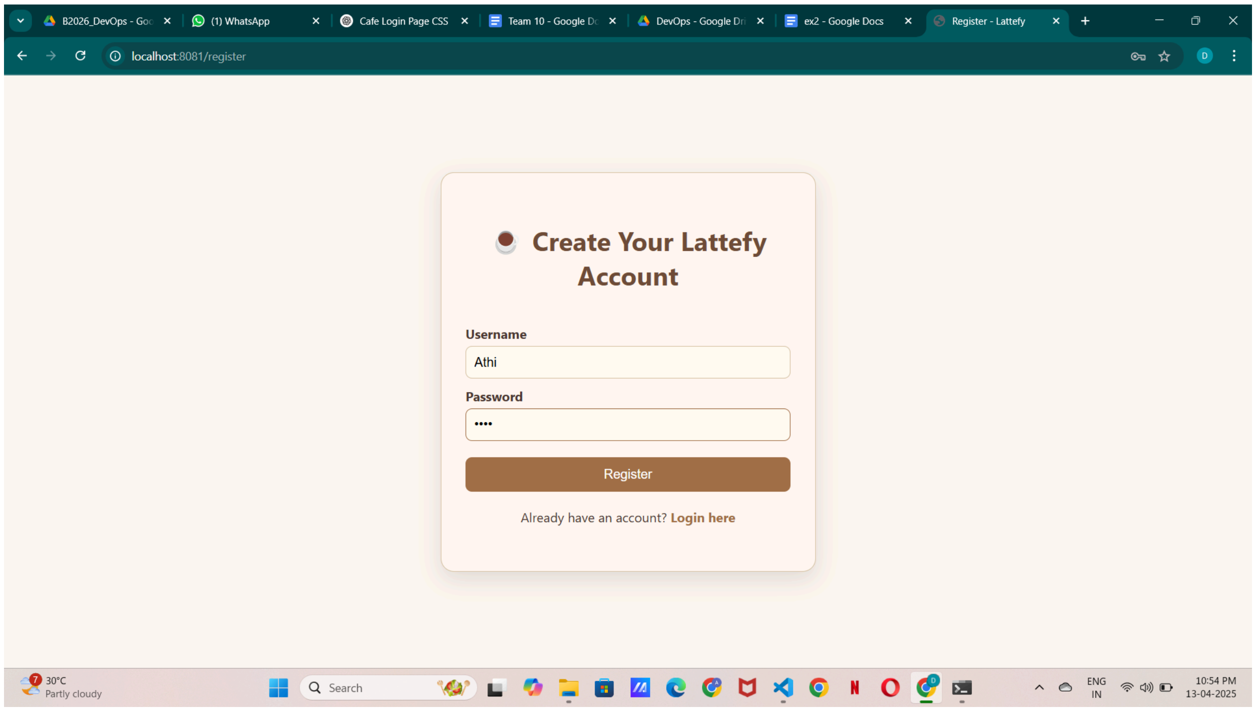
```
package com.batch2.artifact1;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class Artifact1ApplicationTests {

        @Test
        void contextLoads() {
        }

}
```

LoginControllerTest.java:

```
package com.batch2.artifact1;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.service.LoginService;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.test.web.servlet.MockMvc;

import static org.mockito.Mockito.when;
import static org.mockito.ArgumentMatchers.anyString;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
```

```
@SpringBootTest
@AutoConfigureMockMvc
public class LoginControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private LoginService loginService;

    @Test
    public void testSuccessfulLogin() throws Exception {
        Login user = new Login();
        user.setUsername("admin");
        when(loginService.log(anyString(), anyString())).thenReturn(user);

        mockMvc.perform(post("/login")
                .param("username", "admin")
                .param("password", "password"))
                .andExpect(status().is3xxRedirection())
                .andExpect(redirectedUrl("/home"));
    }

    @Test
    public void testFailedLogin() throws Exception {
        when(loginService.log(anyString(), anyString())).thenReturn(null);

        mockMvc.perform(post("/login")
                .param("username", "invalidUser")
                .param("password", "wrongPass"))
                .andExpect(status().isOk())
                .andExpect(view().name("login"))
                .andExpect(model().attributeExists("error"));
    }
}
```

OUTPUT:



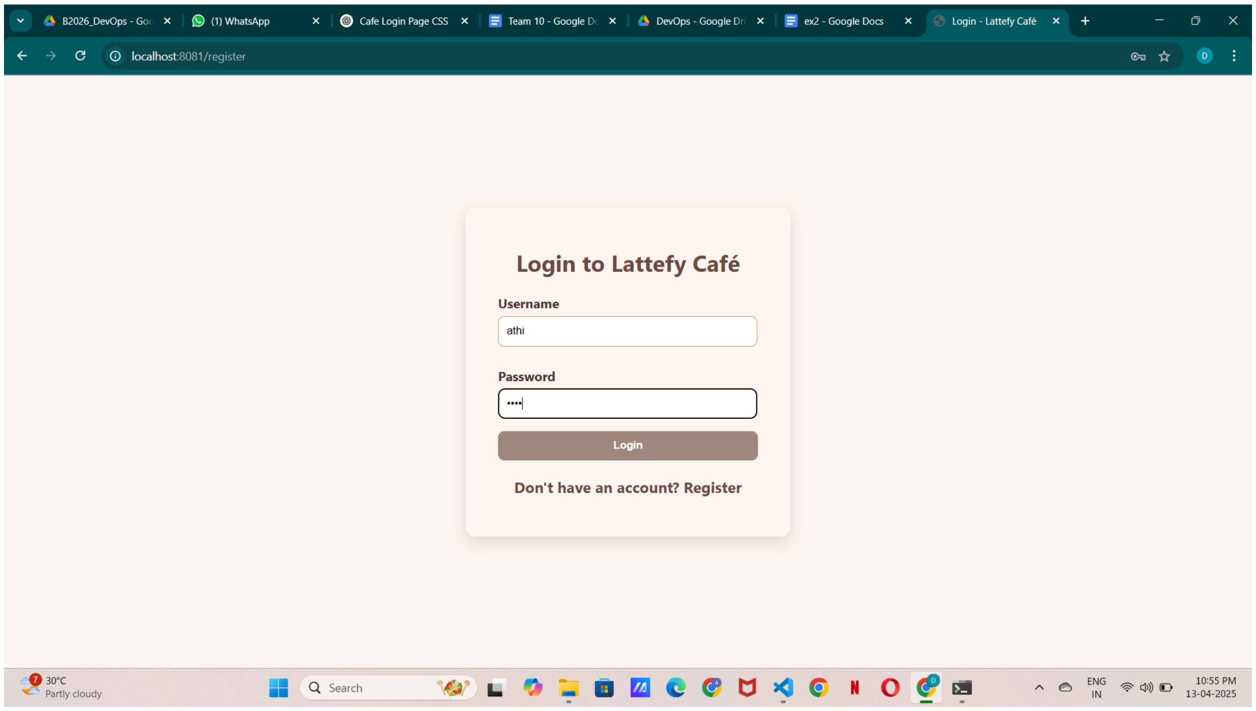
Create Your Lattefy Account

Username

Athi

Password

••••



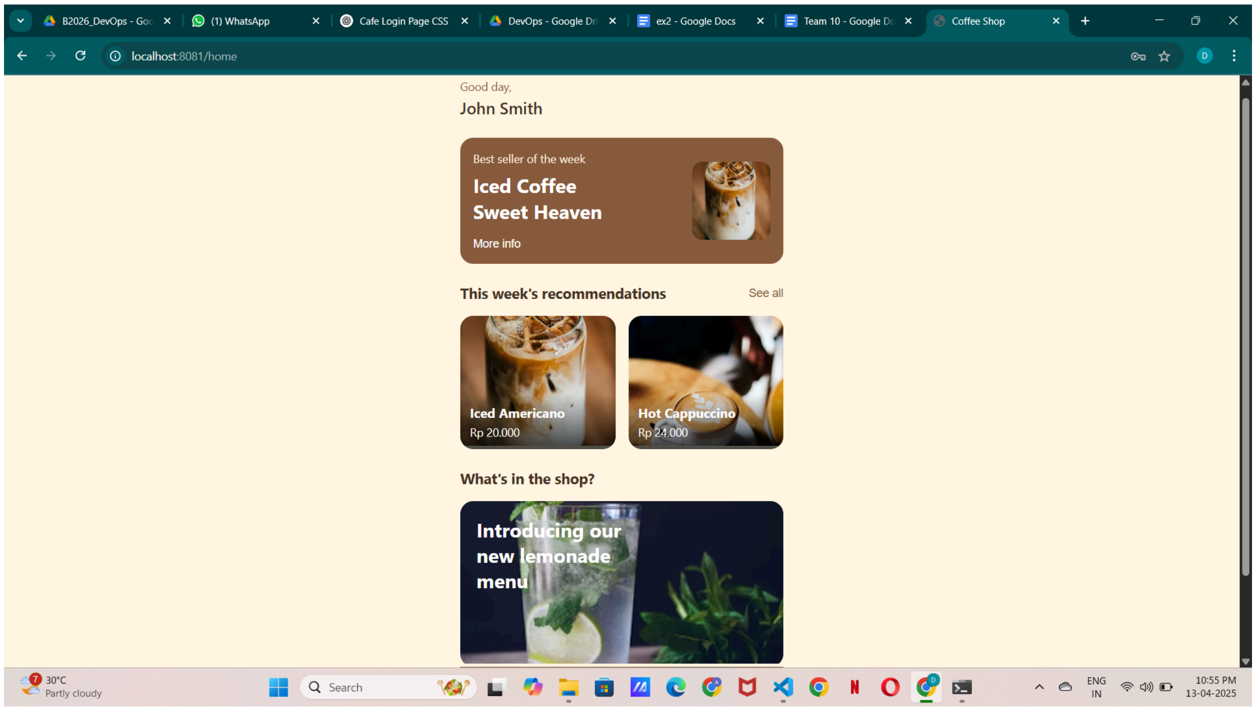
Login to Lattefy Café

Username

athi

Password

••••



Good day,

John Smith

Best seller of the week

Iced Coffee Sweet Heaven

More info

This week's recommendations

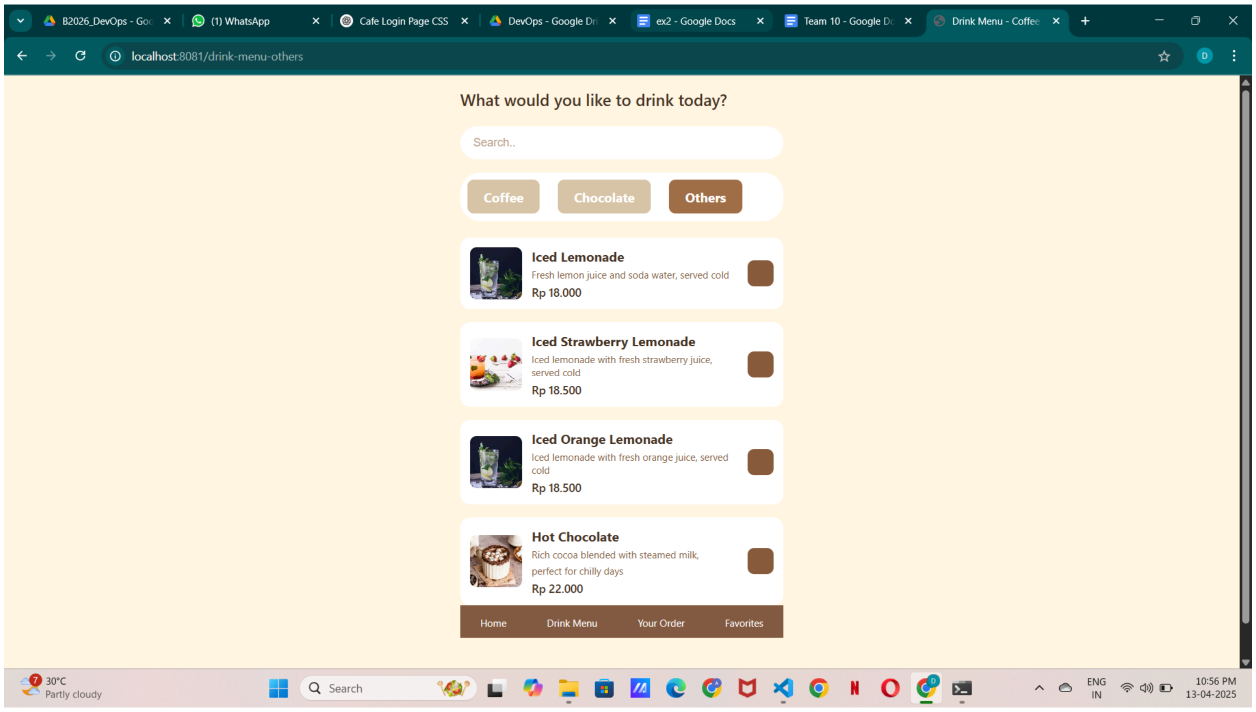
See all

Iced Americano
Rp 20.000

Hot Cappuccino
Rp 24.000

What's in the shop?

Introducing our new lemonade menu



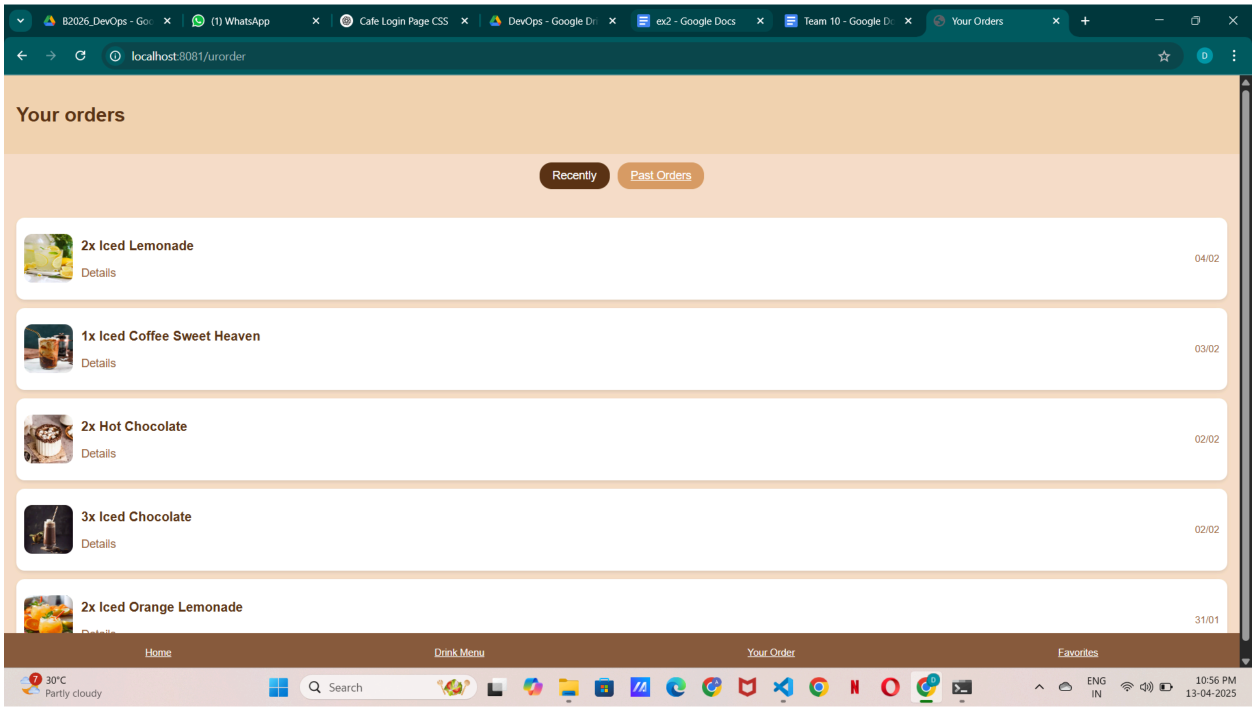
What would you like to drink today?

Iced Lemonade
Fresh lemon juice and soda water, served cold
Rp 18.000

Iced Strawberry Lemonade
Iced lemonade with fresh strawberry juice, served cold
Rp 18.500

Iced Orange Lemonade
Iced lemonade with fresh orange juice, served cold
Rp 18.500

Hot Chocolate
Rich cocoa blended with steamed milk, perfect for chilly days
Rp 22.000


B2026_DevOps - Goo | (1) WhatsApp | Cafe Login Page CSS | DevOps - Google Dri | ex2 - Google Docs | Team 10 - Google Do | Your Orders

localhost:8081/urorder

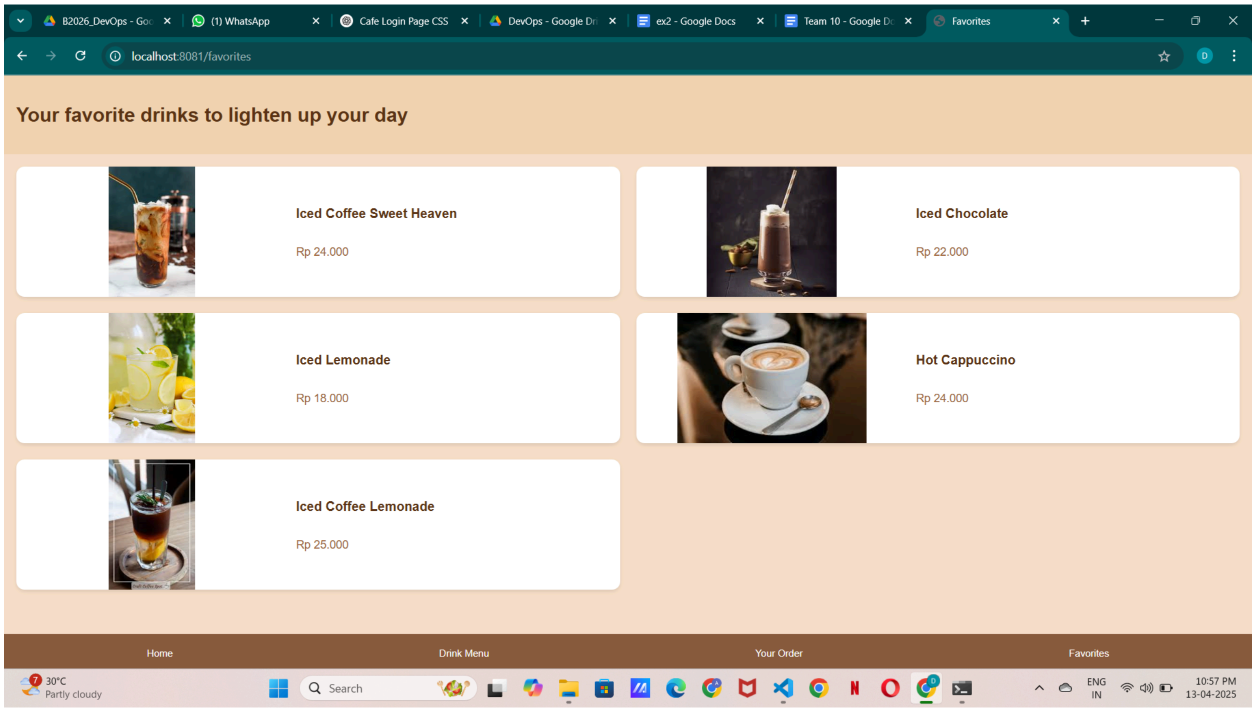
Your orders

Recently | Past Orders

2x Iced Lemonade	Details	04/02
1x Iced Coffee Sweet Heaven	Details	03/02
2x Hot Chocolate	Details	02/02
3x Iced Chocolate	Details	02/02
2x Iced Orange Lemonade	Details	31/01

Home | Drink Menu | Your Order | Favorites

30°C Partly cloudy | Search | ENG IN | 10:56 PM 13-04-2025


B2026_DevOps - Goo | (1) WhatsApp | Cafe Login Page CSS | DevOps - Google Dri | ex2 - Google Docs | Team 10 - Google Do | Favorites

localhost:8081/favorites

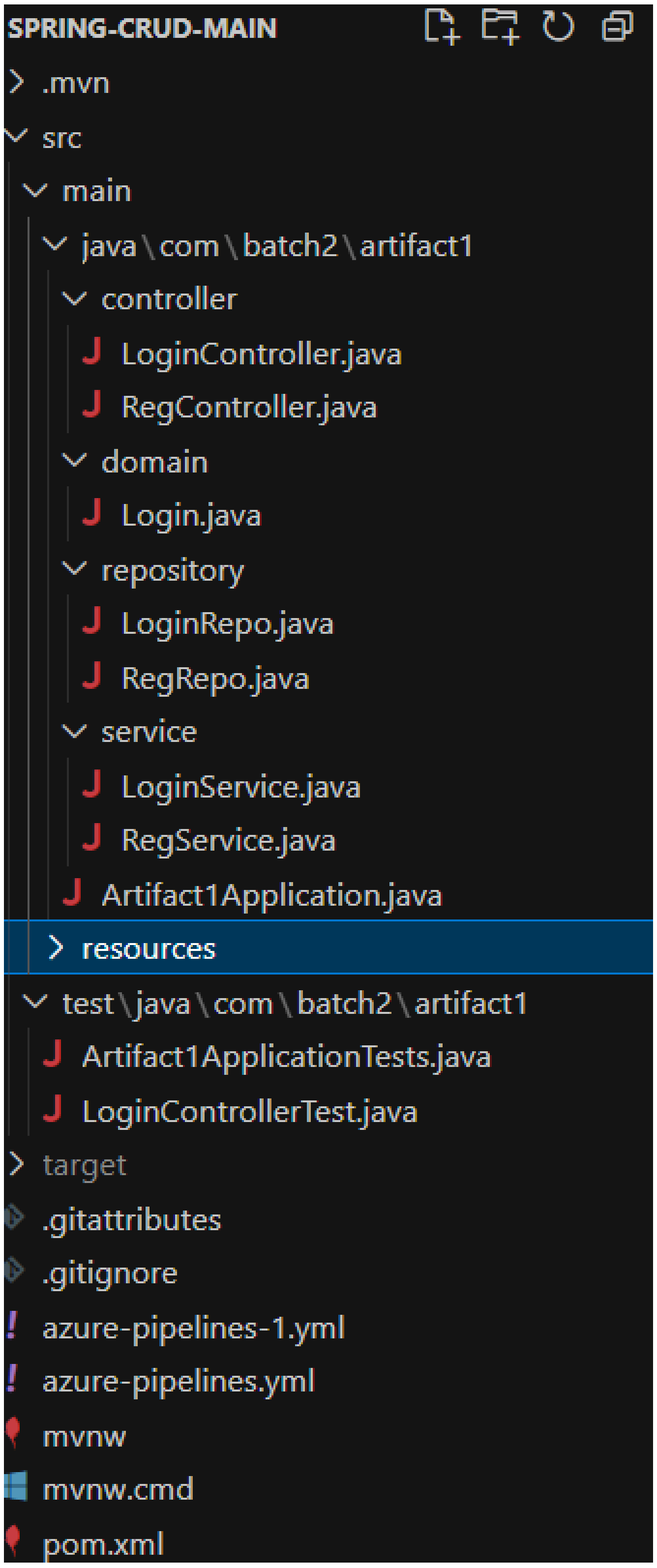
Your favorite drinks to lighten up your day

Iced Coffee Sweet Heaven	Rp 24.000
Iced Chocolate	Rp 22.000
Iced Lemonade	Rp 18.000
Hot Cappuccino	Rp 24.000
Iced Coffee Lemonade	Rp 25.000

Home | Drink Menu | Your Order | Favorites

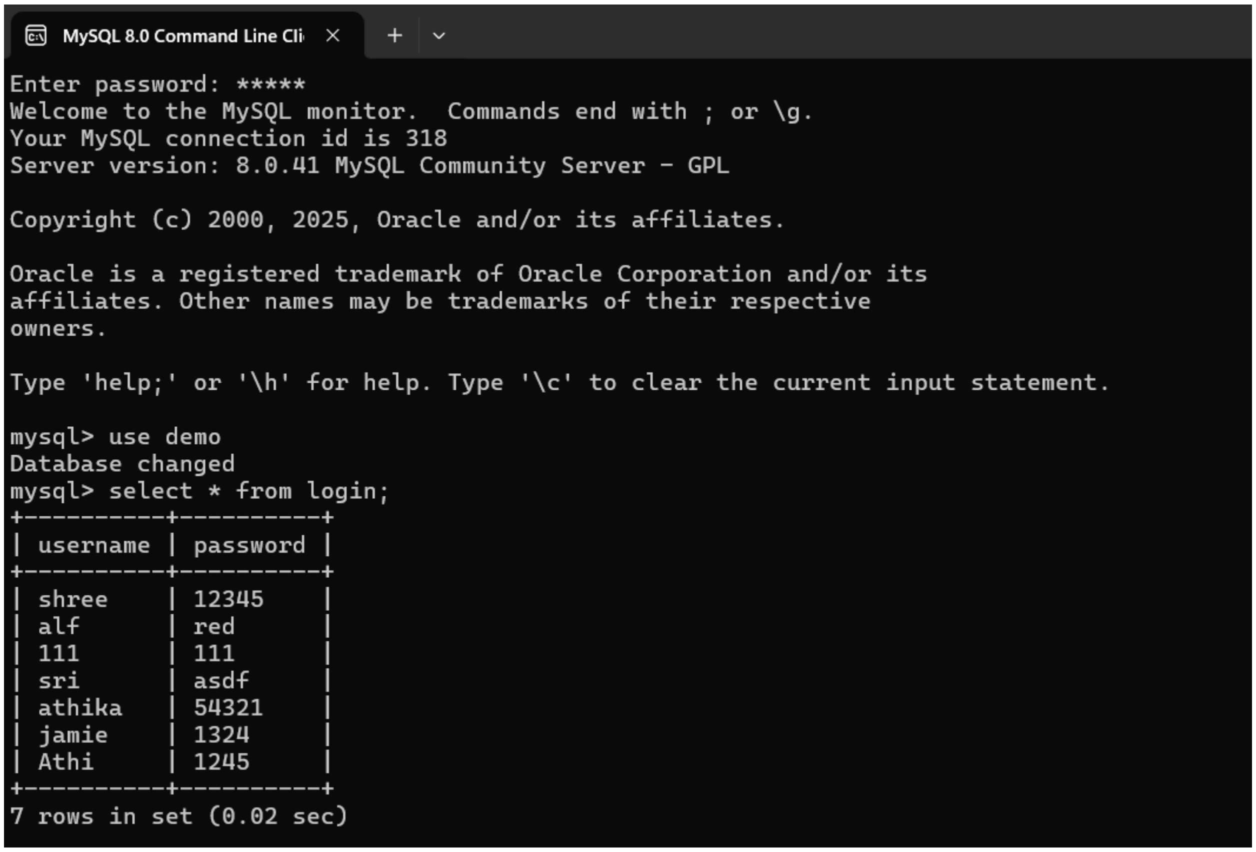
30°C Partly cloudy | Search | ENG IN | 10:57 PM 13-04-2025

IMAGE OF DIRECTORY:



```
SPRING-CRUD-MAIN
> .mvn
v src
  v main
    v java\com\batch2\artifact1
      v controller
          LoginController.java
          RegController.java
      v domain
          Login.java
      v repository
          LoginRepo.java
          RegRepo.java
      v service
          LoginService.java
          RegService.java
        Artifact1Application.java
    > resources
  v test\java\com\batch2\artifact1
      Artifact1ApplicationTests.java
      LoginControllerTest.java
> target
  .gitattributes
  .gitignore
  azure-pipelines-1.yml
  azure-pipelines.yml
  mvnw
  mvnw.cmd
  pom.xml
```

DATABASE:



```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 318
Server version: 8.0.41 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use demo
Database changed
mysql> select * from login;
+----------+----------+
| username | password |
+----------+----------+
| shree    | 12345    |
| alf      | red      |
| 111      | 111      |
| sri      | asdf     |
| athika   | 54321    |
| jamie    | 1324     |
| Athi     | 1245     |
+----------+----------+
7 rows in set (0.02 sec)
```

RESULT:

Thus the cafe management system is designed and implemented using mvc architecture.