P2 Demo Project (Customers & Accounts REST End points)

Step 1: Create a new Spring Starter project in STS

Step 2 : Add dependencies

1. Spring Web
2. Spring boot DevTools
3. Spring data jpa
4. H2 database (in-memory -2mb)
5. Mysql driver
6. Postgresql driver
7. Lombok
8. Thymeleaf

Step 3: Adding External Dependencies

1. Swagger
2. Swagger-ui
3. Cucumber
4. Mockito
5. Junit
6. Selenium

Step 4: Add @EnableSwagger2 annotation to the starter class

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

@SpringBootApplication

@EnableSwagger2

public class P2backendApplication {

public static void main(String[] args) {

SpringApplication.run(P2backendApplication.class, args);

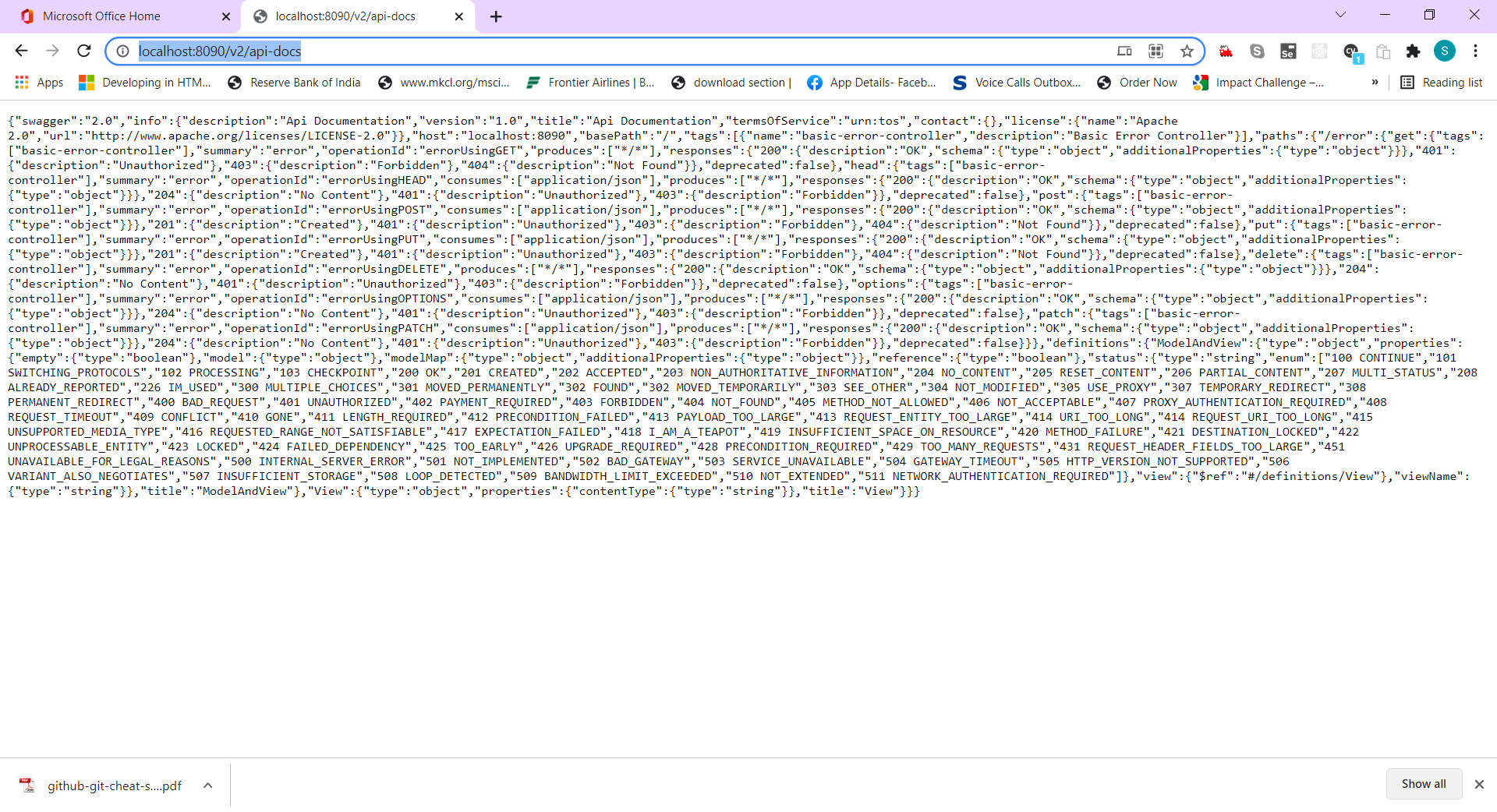
}

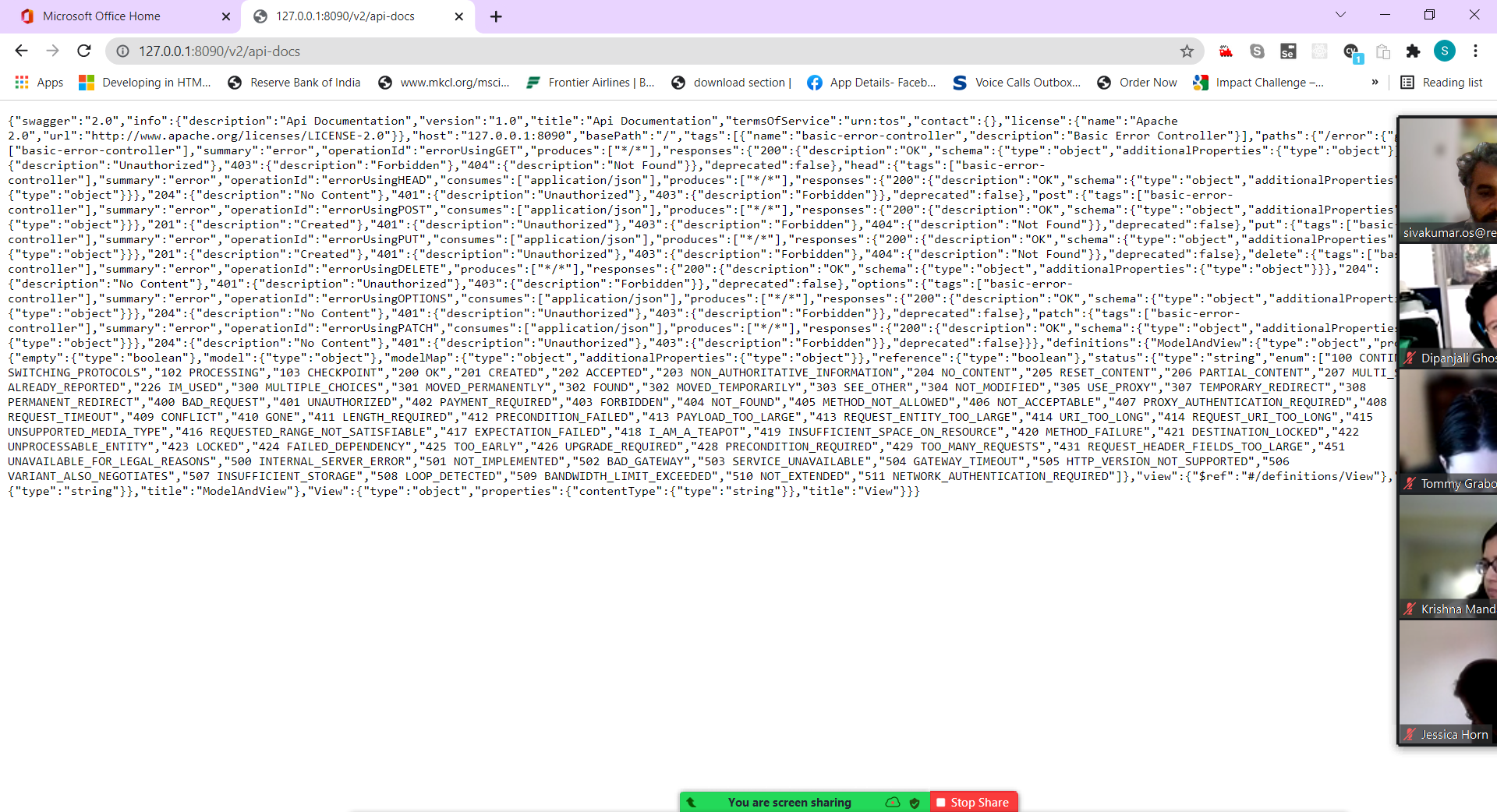
}

Step 5: Run the application and test swagger end points

<http://localhost:8090/v2/api-docs>

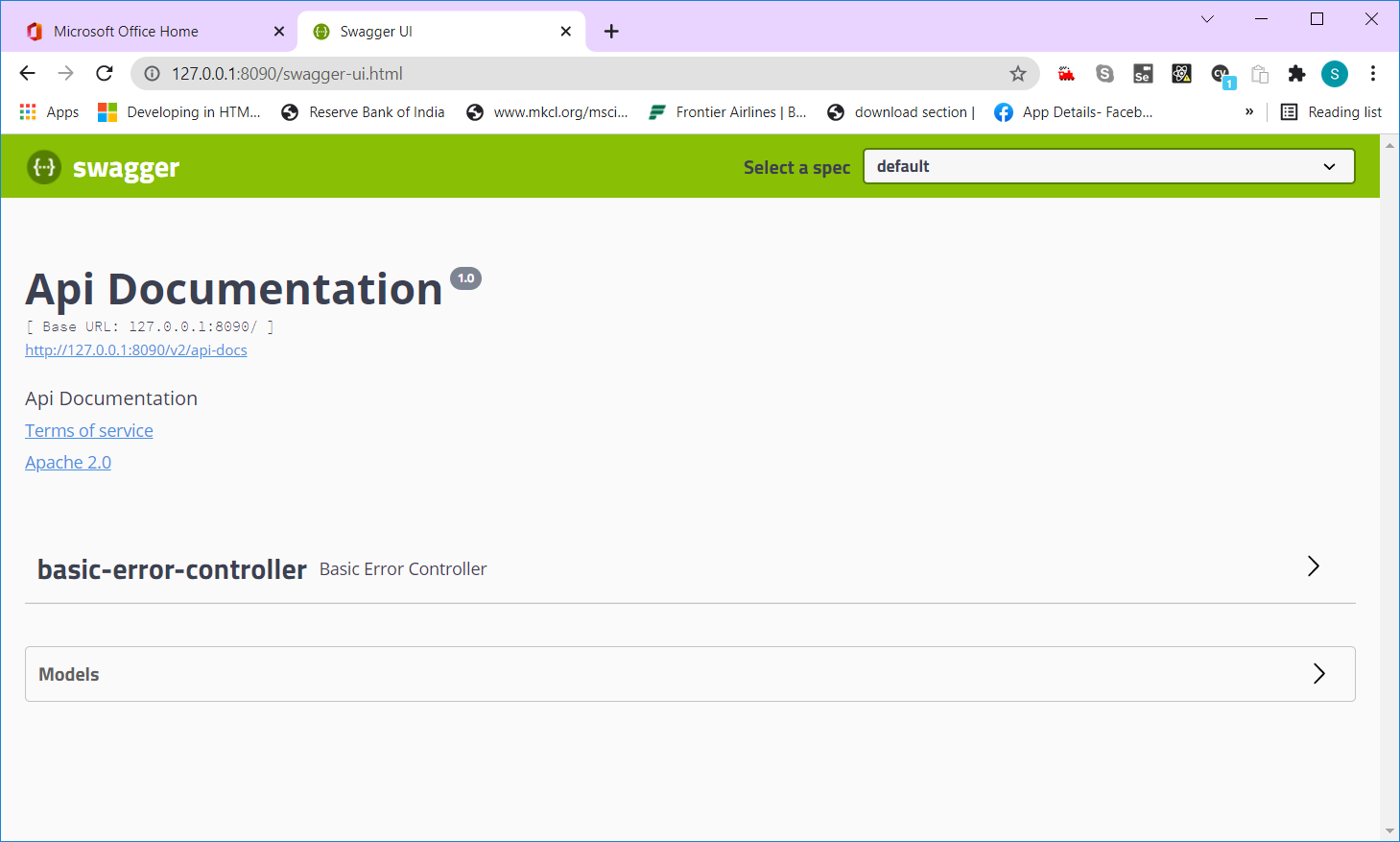
<http://127.0.0.1:8090/v2/api-docs>





Step 6: Checking Swagger ui end point

<http://localhost:8090/swagger-ui.html>



Step 7: Adding DB Configuration in application.properties file

#spring.datasource.url=jdbc:h2:mem:testdb

#spring.datasource.driver-class-name=org.h2.Driver

#spring.datasource.username=sa

#spring.datasource.password=

#spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

#

#spring.h2.console.enabled=true

#spring.datasource.url=jdbc:mysql://localhost:3306/p2

#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#spring.datasource.username=root

#spring.datasource.password=root

#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto= update

spring.jpa.show-sql=true

spring.datasource.url=jdbc:postgresql://localhost:5432/p2

spring.datasource.driver-class-name=org.postgresql.Driver

spring.datasource.username=postgres

spring.datasource.password=postgres

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect

Step 8: Adding Logging config in application.properties file

# Root logger option

log4j.rootLogger=Info, file, stdout

log4j.logger.com.revature=Debug

# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.DailyRollingFileAppender

log4j.appender.file.File=logs/application.log

#log4j.appender.file.MaxFileSize=10MB

#log4j.appender.file.MaxBackupIndex=10

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

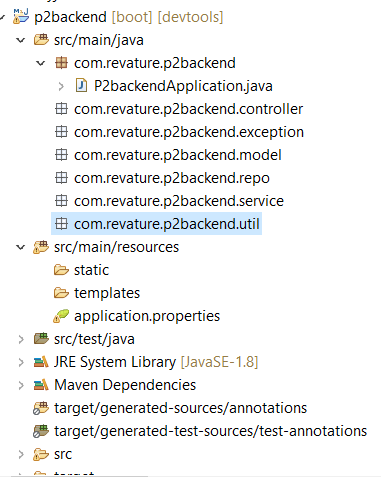
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

log4j.appender.stdout.Target=System.out

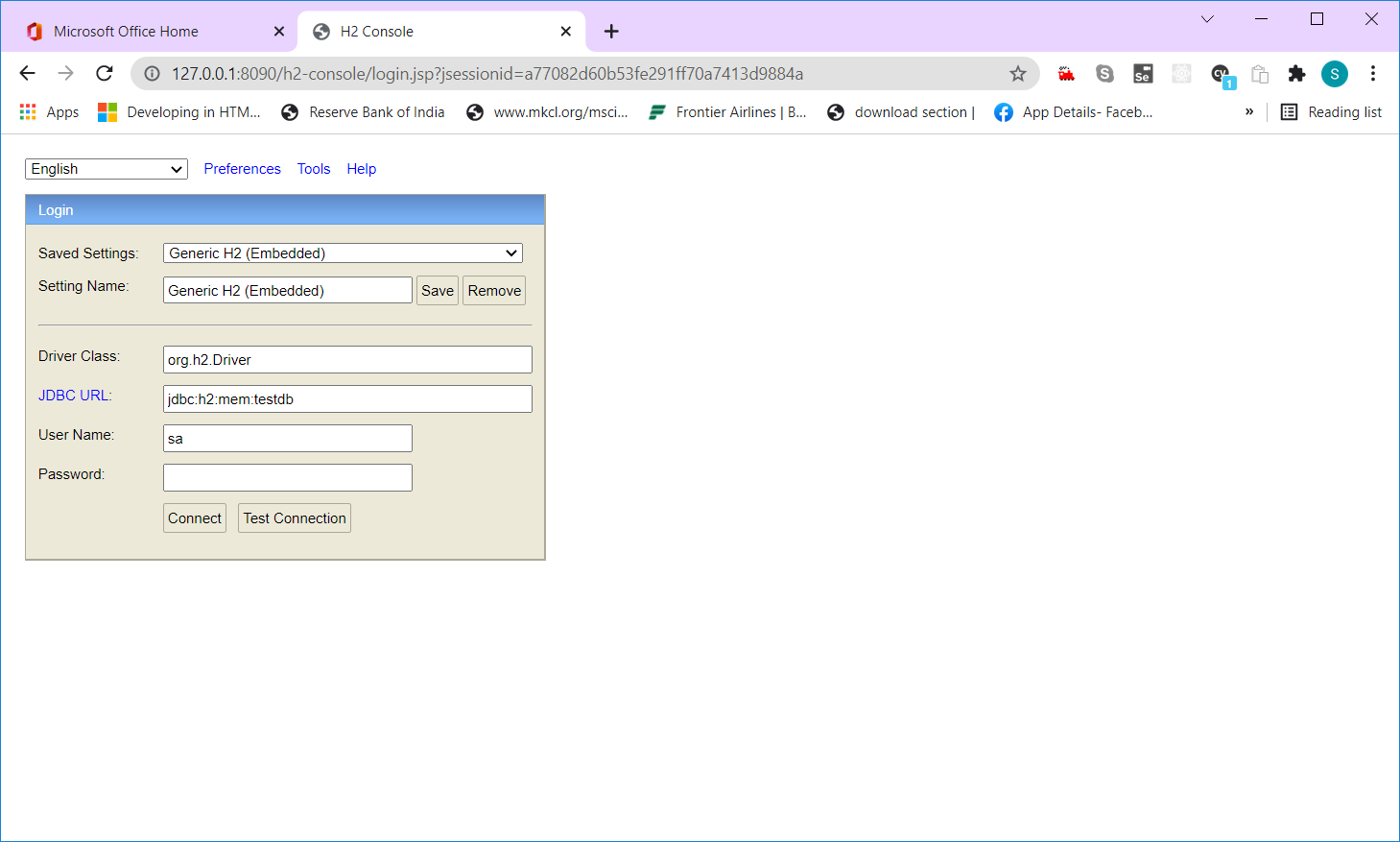
Step 9: Adding the required packages in src/main/java

Controller, exception, model, repo, service, util

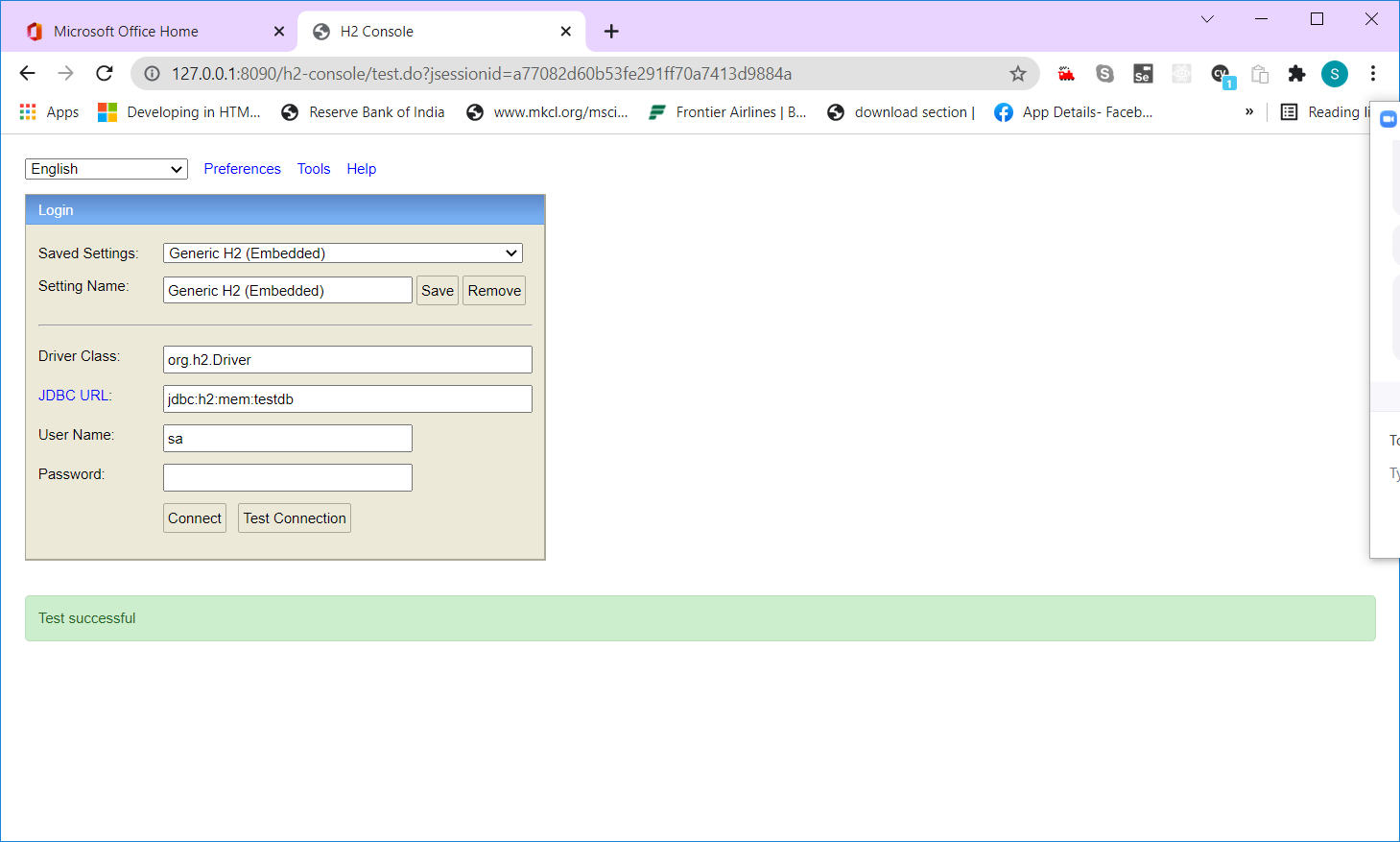


Step 10: Checking the h2-database web console

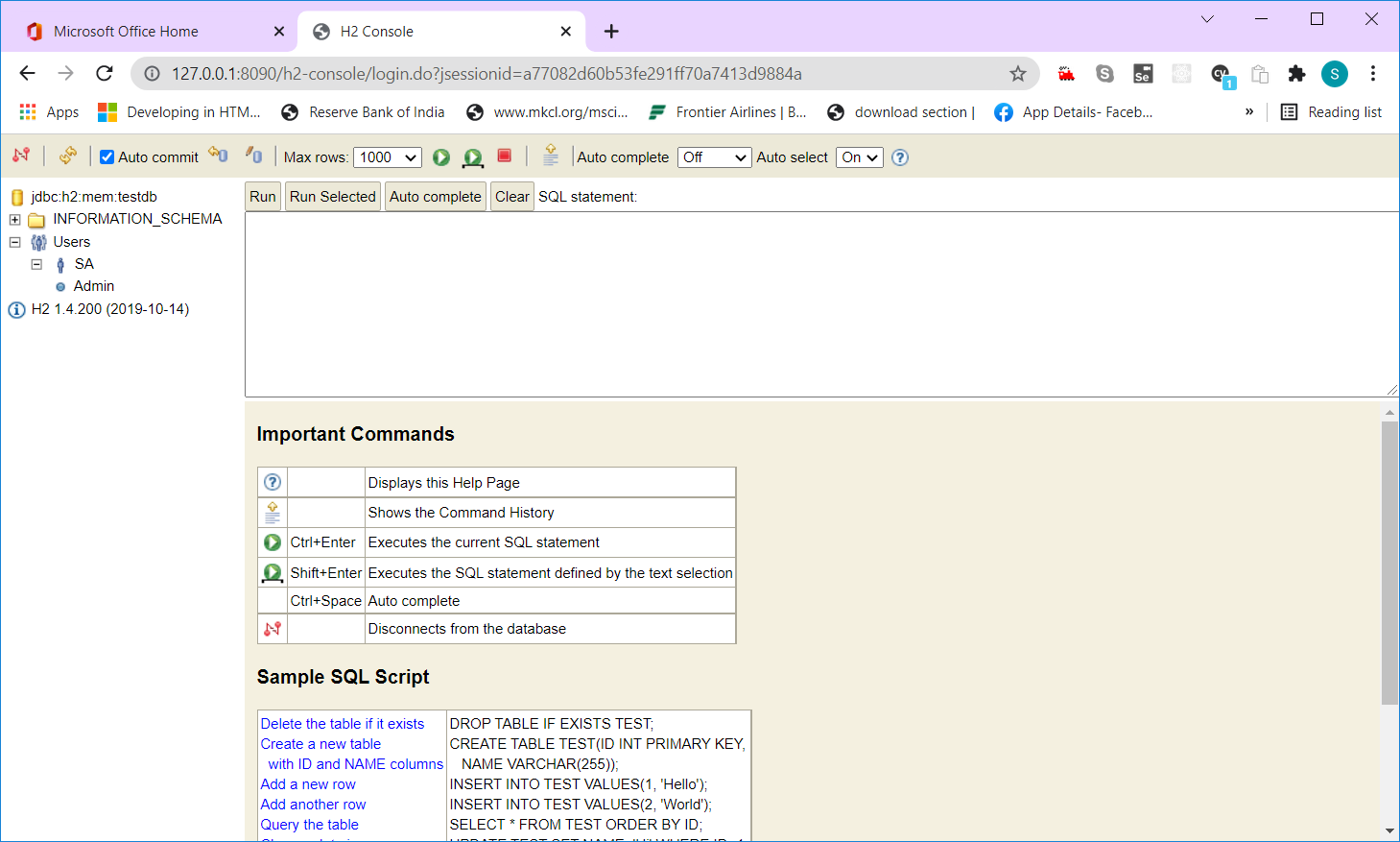
<http://localhost:8090/h2-console> or <http://127.0.0.1:8090/h2-console>



Step 11: Click on “Test Connection” button to check the connection.



Step 12: Click on “Connect” button



Step 13: Add Entity bean class “Customer” inside model package

**package** com.revature.p2backend.model;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** lombok.AllArgsConstructor;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@Entity

@Table(name = "customer")

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@ToString

**public** **class** Customer {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** **int** id;

**private** String name;

**private** String email;

**private** String mobile;

**private** String password;

@Column(name = "is\_active")

**private** **boolean** isActive;

}

Step 14: Add CustomerRepository interface inside repo package

**package** com.revature.p2backend.repo;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.revature.p2backend.model.Customer;

**public** **interface** CustomerRepository **extends** JpaRepository<Customer, Integer>{

**public** List<Customer> findByEmail(String email);

**public** List<Customer> findByMobile(**long** mobile);

}

Step 15: Add CustomerService interface & impl class inside service package

**package** com.revature.p2backend.service;

**import** java.util.List;

**import** com.revature.p2backend.model.Customer;

**public** **interface** CustomerService {

**public** List<Customer> findAll();

**public** Customer findById(**int** id);

**public** List<Customer> findByEmail(String email);

**public** List<Customer> findByMobile(**long** mobile);

**public** **void** save(Customer customer);

**public** **void** update(**int** id, Customer customer);

**public** **void** delete(**int** id);

}

**package** com.revature.p2backend.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.revature.p2backend.model.Customer;

**import** com.revature.p2backend.repo.CustomerRepository;

@Service

**public** **class** CustomerServiceImpl **implements** CustomerService {

@Autowired

CustomerRepository customerRepository;

@Override

**public** List<Customer> findAll() {

**return** customerRepository.findAll();

}

@Override

**public** List<Customer> findByEmail(String email) {

**return** customerRepository.findByEmail(email);

}

@Override

**public** List<Customer> findByMobile(**long** mobile) {

**return** customerRepository.findByMobile(mobile);

}

@Override

**public** Customer findById(**int** id) {

**return** customerRepository.findById(id).get();

}

@Override

**public** **void** save(Customer customer) {

customerRepository.save(customer);

}

@Override

**public** **void** update(**int** id, Customer customer) {

customerRepository.save(customer);

}

@Override

**public** **void** delete(**int** id) {

customerRepository.deleteById(id);

}

}

Step 16: Add CustomerController inside controller package

**package** com.revature.p2backend.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.DeleteMapping;

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

**import** org.springframework.web.bind.annotation.PathVariable;

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

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.revature.p2backend.model.Customer;

**import** com.revature.p2backend.service.CustomerService;

@RestController

**public** **class** CustomerController {

@Autowired

CustomerService customerService;

@GetMapping("/customers")

**public** List<Customer> findAll() {

**return** customerService.findAll();

}

@GetMapping("/customersByEmail/{email}")

**public** List<Customer> findByEmail(@PathVariable String email) {

**return** customerService.findByEmail(email);

}

@GetMapping("/customersByMobile/{mobile}")

**public** List<Customer> findByMobile(@PathVariable **long** mobile) {

**return** customerService.findByMobile(mobile);

}

@GetMapping("/customers/{id}")

**public** Customer findById(@PathVariable **int** id) {

**return** customerService.findById(id);

}

@PostMapping("/customers")

**public** **void** save(Customer customer) {

customerService.save(customer);

}

@PostMapping("/customers/bulk")

**public** **void** save(Customer[] customers) {

**for** (Customer customer : customers) {

customerService.save(customer);

}

}

@PutMapping("/customers/{id}")

**public** **void** update(**int** id, Customer customer) {

customerService.save(customer);

}

@DeleteMapping("/customers/{id}")

**public** **void** delete(**int** id) {

customerService.delete(id);

}

}

Step 17: Add GlobalExceptionHandler class inside exception package

**package** com.revature.p2backend.exception;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.LinkedHashMap;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.stream.Collector;

**import** java.util.stream.Collectors;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.http.converter.HttpMessageNotReadableException;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

**import** org.springframework.web.bind.annotation.ControllerAdvice;

**import** org.springframework.web.context.request.WebRequest;

**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

**import** com.fasterxml.jackson.databind.exc.InvalidFormatException;

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(GlobalExceptionHandler.**class**);

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers, HttpStatus status, WebRequest request) {

***LOGGER***.info("Inside Gloabal Exception Handler Class:handleMethodArgumentNotValid -Start");

Map<String, Object> body = **new** LinkedHashMap<>();

body.put("timestamp", **new** Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult().getFieldErrors().stream().map(x -> x.getDefaultMessage())

.collect(Collectors.*toList*());

body.put("errors", errors);

***LOGGER***.info("Inside Gloabal Exception Handler Class:handleMethodArgumentNotValid - End");

**return** **new** ResponseEntity<>(body,headers, status);

}

@Override

**protected** ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers, HttpStatus status, WebRequest request) {

***LOGGER***.info("Inside Gloabal Exception Handler Class: handleHttpMessageNotReadable -Start");

Map<String, Object> body = **new** LinkedHashMap<>();

body.put("timestamp", **new** Date());

body.put("status", status.value());

body.put("errors", "Bad Request");

// List<String> errors = new ArrayList<>();

// if(ex.getCause() instanceof InvalidFormatException) {

// final Throwable cause = ex.getCause()== null ? ex:ex.getCause();

// for(InvalidFormatException.)

// }

***LOGGER***.info("Inside Gloabal Exception Handler Class:handleHttpMessageNotReadable - End");

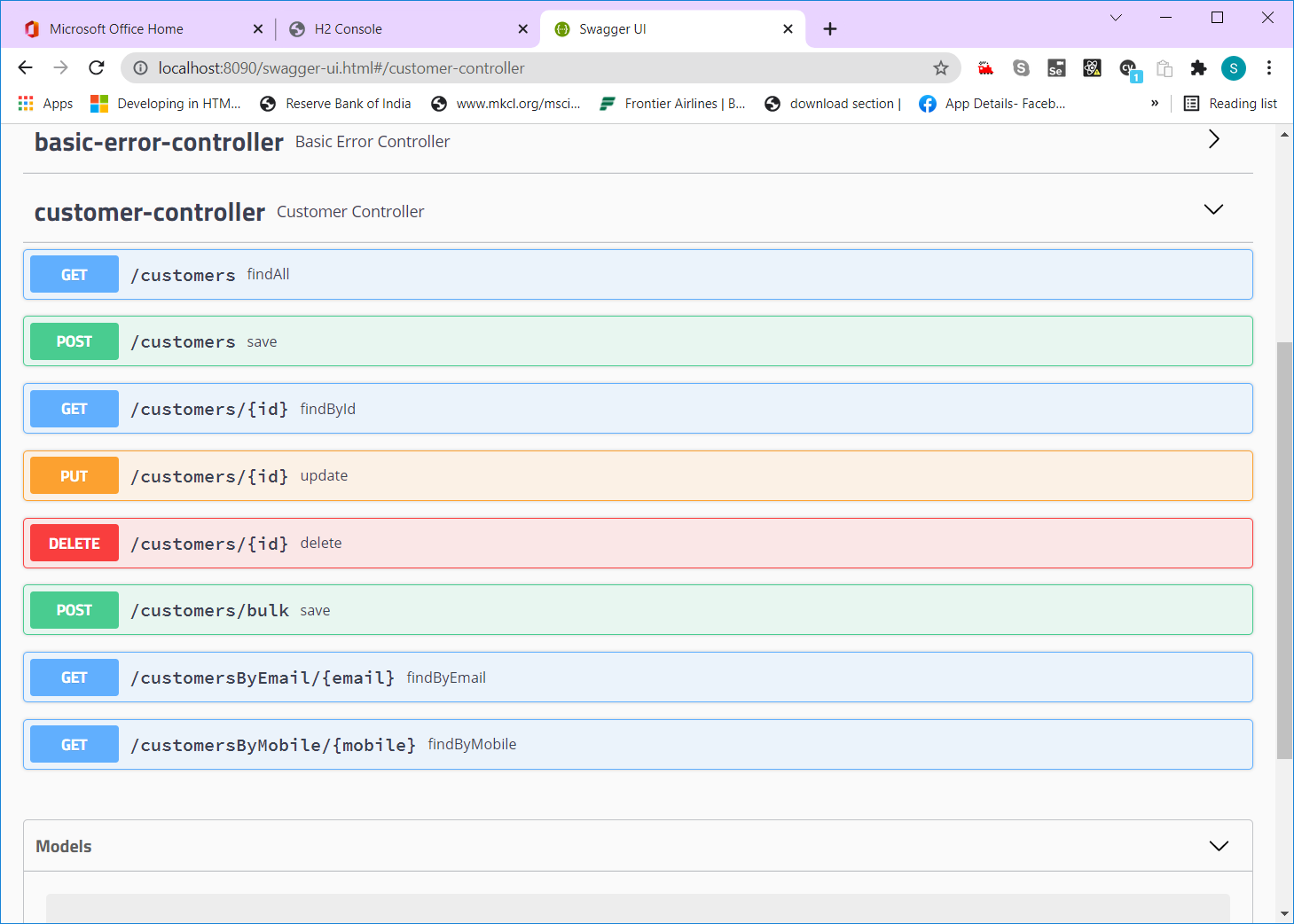
**return** **new** ResponseEntity<>(body,headers, status);

}

}

Step 18: Testing all the end points using swagger2 ui

<http://localhost:8090/swagger-ui.html>



Step 19: Recording Selenium Test cases using Selenium IDE

Step 20: Adding the Front End (Add a new html file “index.html” inside resources/template