ДОДАТОК А

Вихідний код програми

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.authentication.builders.Auth
enticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurit
yConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import javax.sql.DataSource;

@Configuration
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    private static final Logger log = LoggerFactory.getLogger(
SpringSecurityConfig.class );
    private final DataSource dataSource;

    @Autowired
    public SpringSecurityConfig(DataSource dataSource) {
        this.dataSource = dataSource;
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.jdbcAuthentication().dataSource(dataSource)
                .passwordEncoder(new BCryptPasswordEncoder())
                .usersByUsernameQuery("select username,password,enabled
from fms.users where username=?")
                .authoritiesByUsernameQuery("select username,authority from
fms.authorities where username=?");
```

```
        log.info("Spring authentication manager configured.");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http
                .httpBasic()
                .and()
                .authorizeRequests()
                .antMatchers(HttpMethod.POST, "/users/**").permitAll()
                .antMatchers(HttpMethod.GET,
"/clients/**").hasAnyRole("ADMIN","USER")
                .antMatchers("/clients/**").hasRole("ADMIN")

.antMatchers("/users/**").hasRole("ADMIN")
                .and()
                .csrf().disable()
                .formLogin().disable();

        log.info("Spring HttpSecurity configured.");
    }
}

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@EnableSwagger2
@Configuration
public class SwaggerConfig {

    private static final String SWAGGER_API_VERSION = "1.0.0";
```

```
    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .version(SWAGGER_API_VERSION)
                .build();
    }

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .groupName("Fleet Management System")
                .useDefaultResponseMessages(false)
                .select()
                .apis(RequestHandlerSelectors
                        .basePackage("com.nure.diploma.controllers"))
                .paths(PathSelectors.regex("/.*"))
                .build().apiInfo(apiInfo());
    }
}
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.http.    public List<Client> gPage(@RequestParam
int page, @RequestParam int size) {
        log.debug("Getting a page of clients : {} with size : {}", page,
size);

        Pageable pageable = PageRequest.of(page, size,
Sort.by("clientId"));
        return clientService.getPage(pageable).getContent();
    }

    @ResponseStatus(OK)
    @GetMapping(value = "/{id}")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
```

```
    public Client gById(@PathVariable("id") Long id) {
        log.debug("Getting a client with id : {}", id);

        return clientService.getById(id);
    }

    @ResponseStatus(OK)
    @GetMapping(value = "/gByFinger")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public Client getClitByF(@RequestParam("fingerprint") String
fingerprint) {
        log.debug("Setting driver with id : {} to a truck with id : {}");

        return clientService.getByFinger(fingerprint);
    }

    @ResponseStatus(OK)
    @PutMapping(value = "/{id}")
    @ApiResponses(value = {
            @ApiResponse(code = 400, message = "Bad Request"),
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public void upd(@RequestBody Client client, @PathVariable("id") Long
id) {
        log.debug("Updating client with id : {}", id);

        client.setClientId(id);
        clientService.update(client);
    }

    @ResponseStatus(NO_CONTENT)
    @DeleteMapping(value = "/{id}")
    @ApiResponses(value = {
ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.net.URI;
import java.util.List;
```

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

@RestController
@RequestMapping("/cli")
@Api(value = "/cli")
public class ClientsCon {

    private static final Logger log =
LoggerFactory.getLogger(ClientsCon.class);
    private final ClientService clientService;

    public ClientsCon(ClientService clientService) {
        this.clientService = clientService;
    }

    @ResponseStatus(CREATED)
    @PostMapping
    @ApiResponses(value = {
            @ApiResponse(code = 400, message = "Bad Request"),
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden")})
    public @ResponseBody ResponseEntity sv(@RequestBody Client client,
HttpServletRequest request) {
        log.debug("Saving a client");

        long id = clientService.save(client);
        URI uri = URI.create(String.format("%s%d",
request.getRequestURL().toString(), id));
        return ResponseEntity.created(uri).build();
    }

    @ResponseStatus(OK)
    @GetMapping
    @ApiResponses(value = {
            @ApiResponse(code = 204, message = "No Content", response =
Object.class),
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 500, message = "Internal Server Error")})
@ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found"),
```

```
            @ApiResponse(code = 409, message = "Conflict")})
    public void del(@PathVariable("id") Long id) {
        log.debug("Deleting client with id : {}", id);

        clientService.delete(id);
    }
}

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.net.URI;
import java.util.List;

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

@RestController
@RequestMapping("/usr")
@Api(value = "/usr")
public class UsrCon {

    private static final Logger log =
LoggerFactory.getLogger(UsrCon.class);
    private final UserService userService;

    public UsrCon(UserService userService) {
        this.userService = userService;
    }

    @ResponseStatus(CREATED)
    @PostMapping
    @ApiResponses(value = {
            @ApiResponse(code = 400, message = "Bad Request"),
```

```
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 409, message = "Conflict")})
    public @ResponseBody ResponseEntity sv(@RequestBody User user,
HttpServletRequest request) {
        log.debug("Saving a user");

        Long id = userService.save(user);
        URI uri = URI.create(String.format("%s%d",
request.getRequestURL().toString(), id));
        return ResponseEntity.created(uri).build();
    }

    @ResponseStatus(OK)
    @PutMapping(value = "/sVal")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public void sVal(@RequestParam("id") Long id) {
        log.debug("Setting admin rights to a user with id : {}", id);

        userService.setAdmin(id);
    }

    @ResponseStatus(OK)
    @PutMapping(value = "/sUnVal")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public void sUnVal(@RequestParam("id") Long id) {
        log.debug("Setting user rights to a user with id : {}", id);

        userService.setUser(id);
    }

    @ResponseStatus(OK)
    @PutMapping(value = "/unvalidate")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found"),
```

```
        @ApiResponse(code = 409, message = "Conflict")})
public void unvalidate(@RequestParam("id") Long id) {
    log.debug("Banning a user with id : {}", id);

    userService.ban(id);
}

@ResponseStatus(OK)
@PutMapping(value = "/validate")
@ApiResponses(value = {
        @ApiResponse(code = 401, message = "Unauthorized"),
        @ApiResponse(code = 403, message = "Forbidden"),
        @ApiResponse(code = 404, message = "Not Found")})
public void validate(@RequestParam("id") Long id) {
    log.debug("Removing ban from a user with id : {}", id);

    userService.unBan(id);
}

@ResponseStatus(NO_CONTENT)
@DeleteMapping(value = "/{id}")
@ApiResponses(value = {
        @ApiResponse(code = 401, message = "Unauthorized"),
        @ApiResponse(code = 403, message = "Forbidden"),
        @ApiResponse(code = 404, message = "Not Found"),
        @ApiResponse(code = 409, message = "Conflict")})
public void del(@PathVariable("id") Long id) {
    log.debug("Deleting user with id : {}", id);

    userService.delete(id);
}

@ResponseStatus(OK)
@GetMapping(value = "/gByUsrnm/{username}")
@ApiResponses(value = {
        @ApiResponse(code = 401, message = "Unauthorized"),
        @ApiResponse(code = 403, message = "Forbidden"),
        @ApiResponse(code = 404, message = "Not Found")})
public User gByUsrnm(@PathVariable("username") String username) {
    log.debug("Getting a user with username : {}", username);

    return userService.getByUsername(username);
}
```

```
    @ResponseStatus(OK)
    @GetMapping(value = "/gAuthorByUsrnm/{username}")
    @ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public Authority gAuthorByUsrnm(@PathVariable("username") String
username) {
        log.debug("Getting a user with username : {}", username);

        return userService.findAuthoritiesByUsername(username);
    }

    @ResponseStatus(OK)
    @GetMapping(value = "/{id}")
@ApiResponses(value = {
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 404, message = "Not Found")})
    public User gById(@PathVariable("id") Long id) {
        log.debug("Getting a user with id : {}", id);

        return userService.getById(id);
    }

    @ResponseStatus(OK)
    @GetMapping
    @ApiResponses(value = {
            @ApiResponse(code = 204, message = "No Content", response =
Object.class),
            @ApiResponse(code = 401, message = "Unauthorized"),
            @ApiResponse(code = 403, message = "Forbidden"),
            @ApiResponse(code = 500, message = "Internal Server Error")})
    public List<User> gPage(@RequestParam int page, @RequestParam int size)
{
        log.debug("Getting a page of users : {} with size : {}", page,
size);

        Pageable pageable = PageRequest.of(page, size, Sort.by("userId"));
        return userService.getPage(pageable).getContent();
    }
}
```

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.transaction.annotation.Transactional;


@Transactional
public interface UsrServ {

    Long sv(User user);

    void unvalidate(Long id);

    void validate(Long id);

    void del(Long id);

    @Transactional(readOnly = true)
    Authority gAuthorByUsrnm(String username);

    @Transactional(readOnly = true)
    User gByUsrnm(String username);

    @Transactional(readOnly = true)
    User gById(Long id);
    @Transactional(readOnly = true)
    Page<User> getPage(Pageable pageable);

    void ban(Long id);

    void unBan(Long id);
}
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

@Service
public class UsrServImpl implements UsrServ {
```

```
    private static final Logger log =
LoggerFactory.getLogger(UsrServImpl.class);
    private final UserRepository userRepository;
    private final AuthorityRepository authorityRepository;
    private final UserValidatorService userValidatorService;

    @Autowired
    public UsrServImpl(UserRepository userRepository, AuthorityRepository
authorityRepository,
                           UserValidatorService userValidatorService) {
        this.userRepository = userRepository;
        this.authorityRepository = authorityRepository;
        this.userValidatorService = userValidatorService;
    }

    @Override
    public Long sv(User user) {
        userValidatorService.validateFields(user);
        userValidatorService.validateUniqueFields(user);
        user.setPassword(new
BCryptPasswordEncoder().encode(user.getPassword()));
        user.setEnabled(true);
        User savedUser = userRepository.save(user);
        authorityRepository.save(new Authority(user.getUsername(),
"ROLE_USER"));

        log.info("Successfully saved a route with id : {}",
savedUser.getUserId());

        return savedUser.getUserId();
    }

    @Override
    public void validate(Long id) {
User user = userRepository.findById(id)
                .orElseThrow(() -> new NoEntityFoundException(User.class,
id));
        userValidatorService.isEntityExists(User.class, id);
        authorityRepository.save(new Authority(user.getUsername(),
"ROLE_ADMIN"));

        log.info("Successfully set admin rights to user with id : {}", id);
    }
```

```
    @Override
    public void unvalidate(Long id) {
        User user = userRepository.findById(id)
                .orElseThrow(() -> new NoEntityFoundException(User.class,
id));
        userValidatorService.isEntityExists(User.class, id);
        authorityRepository.save(new Authority(user.getUsername(),
"ROLE_USER"));

        log.info("Successfully set user rights to user with id : {}", id);
    }

    @Override
    public void del(Long id) {
        User user = userRepository.findById(id)
                .orElseThrow(() -> new NoEntityFoundException(User.class,
id));
        userValidatorService.isEntityExists(User.class, id);
        userValidatorService.isUserNotAdmin(id);
        authorityRepository.deleteById(user.getUsername());
        userRepository.deleteById(id);

        log.info("Successfully deleted a user with id : {}", id);
    }

    @Override
    public Authority gAuthorByUsrnm(String username) {
        userValidatorService.isEntityExists(Authority.class, username);
        Authority authority = authorityRepository.findById(username)
                .orElseThrow(() -> new
NoEntityFoundException(Authority.class, username));

        log.info("Successfully got user's : {} authority", username);

        return authority;
    }

    @Override
    public User gByUsrnm(String username) {
        User user = userRepository.findByUsername(username)
                .orElseThrow(() -> new NoEntityFoundException(User.class,
username, "username"));
```

```
        log.info("Got a user by its username : {}", username);
return user;
    }

    @Override
    public User gById(Long id) {
        User user = userRepository.findById(id).orElseThrow(() -> new
NoEntityFoundException(User.class, id));

        log.info("Got a user by id : {}", id);

        return user;
    }

    @Override
    public Page<User> gPage(Pageable pageable) {
        Page<User> users = userRepository.findAll(pageable);

        log.info("Got a page with users number : {}, size : {}",
pageable.getPageNumber(), pageable.getPageSize());

        return users;
    }

    @Override
    public void sVal(Long id) {
        userValidatorService.isEntityExists(User.class, id);
        userValidatorService.isUserNotAdmin(id);
        User user = userRepository.findById(id)
                .orElseThrow(() -> new NoEntityFoundException(User.class,
id));
        user.setEnabled(false);
        userRepository.save(user);

        log.info("Successfully banned user with id : {}", id);
    }

    @Override
    public void sUnVal(Long id) {
        userValidatorService.isEntityExists(User.class, id);
        User user = userRepository.findById(id)
```

```
                .orElseThrow(() -> new NoEntityFoundException(User.class,
id));
        user.setEnabled(true);
        userRepository.save(user);

        log.info("Successfully unbanned user with id : {}", id);
    }
}
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class Application extends SpringBootServletInitializer {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}    public void navigateToAuthenticationView() {
        startActivity(new Intent(getApp().getApplicationContext(),
AuthenticationActivity.class));
        if (getActivity() != null) {
            getActivity().finish();
        }
    }

import android.Manifest;
import android.app.Instrumentation;
import android.content.Context;
import android.content.pm.PackageManager;
import android.hardware.fingerprint.FingerprintManager;
import android.os.CancellationSignal;
import android.support.v4.app.ActivityCompat;
import android.widget.Toast;


import com.google.gson.Gson;
import com.google.gson.JsonObject;

import org.json.JSONObject;

import java.io.IOException;
import java.util.concurrent.TimeUnit;
```

```
import okhttp3.Credentials;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;


public class FingerprintHandler extends
FingerprintManager.AuthenticationCallback {

    private CancellationSignal cancellationSignal;
    private Context context;
    private String responseMsg = null;
    private String userName;
    private String password;
    private double price;


    public FingerprintHandler(Context mContext) {
        context = mContext;
    }

    public void setPass(String pass) {
        password = pass;
    }

    public void setUser(String user) {
        userName = user;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void startAuth(FingerprintManager manager,
FingerprintManager.CryptoObject cryptoObject) {
        cancellationSignal = new CancellationSignal();
        if (ActivityCompat.checkSelfPermission(context,
Manifest.permission.USE_FINGERPRINT) != PackageManager.PERMISSION_GRANTED)
{
            return;
```

```
        }
        manager.authenticate(cryptoObject, cancellationSignal, 0, this,
null);

    }

    @Override
    public void onAuthenticationError(int errMsgId,
                                      CharSequence errString) {
        Toast.makeText(context,
                "Authentication error\n" + errString,
                Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAuthenticationFailed() {
        Toast.makeText(context,
                "Authentication failed",
                Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAuthenticationHelp(int helpMsgId,
                                     CharSequence helpString) {
        Toast.makeText(context,
                "Authentication help\n" + helpString,
                Toast.LENGTH_LONG).show();

    }

    public static Instrumentation callLifeCycleMethod() {
        return new Instrumentation();
    }

    @Override
    public void onAuthenticationSucceeded(
            FingerprintManager.AuthenticationResult result) {

        thread.start();
        while (responseMsg == null) {
            try {
                TimeUnit.MILLISECONDS.sleep(10);
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
            }
        }
        Toast.makeText(context,
                " Authorization successful! \n" + responseMsg,
                Toast.LENGTH_LONG).show();
        responseMsg = null;
    }

    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                OkHttpClient client = new OkHttpClient();
                String userFingerprint =
"$2a$10$0lE0tutEI9wMQX3QRfXSvuvVodRbcZ42qQdNZ.a1JStowk/wYre8.";
                String url =
"http://192.168.0.102:8080/fms/clients/getByFinger?fingerprint=" +
userFingerprint;
                String url2 = "http://192.168.0.102:8080/fms/clients/3";
                String credential = Credentials.basic(userName, password);
                String clintFomDB = null;
                final Request request = new Request.Builder()
                        .url(url)
                        .header("Authorization", credential)
                        .build();
                try {
                    Response response = client.newCall(request).execute();
                    clintFomDB = response.body().string();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                Gson gson = new Gson();
                JsonObject clientJson = gson.fromJson(clintFomDB,
JsonObject.class);
                double result = clientJson.get("amount").getAsDouble();
                clientJson.addProperty("amount", result - price);

                final MediaType JSON
                        = MediaType.parse("application/json; charset=utf-
8");
```

```
                RequestBody body = RequestBody.create(JSON,
clientJson.toString());
                final Request request2 = new Request.Builder()
                        .url(url2)
                        .header("Authorization", credential)
                        .put(body)
                        .build();
                try {
                    Response response = client.newCall(request2).execute();
                    if (response.isSuccessful()) {
                        responseMsg = "Transaction successful.";
                    } else responseMsg = "Error while performing
transaction.";
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });

}
import android.Manifest;
import android.app.KeyguardManager;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.hardware.fingerprint.FingerprintManager;
import android.os.Build;
import android.os.Bundle;
import android.security.keystore.KeyGenParameterSpec;
import android.security.keystore.KeyPermanentlyInvalidatedException;
import android.security.keystore.KeyProperties;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
```

```
import java.security.NoSuchProviderException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

public class MainActivity extends AppCompatActivity {

    private static final String KEY_NAME = "yourKey";
    private Cipher cipher;
    private KeyStore keyStore;
    private KeyGenerator keyGenerator;
    private TextView textView, startTextView;
    private FingerprintManager.CryptoObject cryptoObject;
    private FingerprintManager fingerprintManager;
    private KeyguardManager keyguardManager;
    private String username;
    private String password;
    private double price;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        startTextView = (TextView) findViewById(R.id.startView);

        Intent intent = getIntent();
        username = intent.getStringExtra("username");
        password = intent.getStringExtra("password");
        price = Double.parseDouble(intent.getStringExtra("price"));
        String route = intent.getStringExtra("route");

        startTextView.setText("You are on the route №" + route +
                "\n One ticket costs " + price + "$");
    }

    @Override
    protected void onResume() {
        super.onResume();
```

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
            keyguardManager =
                    (KeyguardManager) getSystemService(KEYGUARD_SERVICE);
            fingerprintManager =
                    (FingerprintManager)
getSystemService(FINGERPRINT_SERVICE);

            textView = (TextView) findViewById(R.id.textview);
            if (!fingerprintManager.isHardwareDetected()) {
                textView.setText("Your device doesn't support fingerprint
authentication");
            }
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.USE_FINGERPRINT) != PackageManager.PERMISSION_GRANTED)
{
                textView.setText("Please enable the fingerprint
permission");
            }
            if (!fingerprintManager.hasEnrolledFingerprints()) {
                textView.setText("No fingerprint configured. Please
register at least one fingerprint in your device's Settings");
            }
            if (!keyguardManager.isKeyguardSecure()) {
                textView.setText("Please enable lockscreen security in your
device's Settings");
            } else {
                try {
                    generateKey();
                } catch (FingerprintException e) {
                    e.printStackTrace();
                }
                if (initCipher()) {
                    cryptoObject = new
FingerprintManager.CryptoObject(cipher);
                    FingerprintHandler helper = new
FingerprintHandler(this);
                    helper.setPass(password);
                    helper.setUser(username);
                    helper.setPrice(price);
                    helper.startAuth(fingerprintManager, cryptoObject);

                    //TODO повторять это в authSuccess
                }
```

```
            }
        }
    }

    private void generateKey() throws FingerprintException {
        try {
            keyStore = KeyStore.getInstance("AndroidKeyStore");
            keyGenerator =
KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
"AndroidKeyStore");
            keyStore.load(null);
            keyGenerator.init(new
                    KeyGenParameterSpec.Builder(KEY_NAME,
                    KeyProperties.PURPOSE_ENCRYPT |
                            KeyProperties.PURPOSE_DECRYPT)
                    .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                    .setUserAuthenticationRequired(true)
                    .setEncryptionPaddings(
                            KeyProperties.ENCRYPTION_PADDING_PKCS7)
                    .build());
            keyGenerator.generateKey();
        } catch (KeyStoreException
                | NoSuchAlgorithmException
                | NoSuchProviderException
                | InvalidAlgorithmParameterException
                | CertificateException
                | IOException exc) {
            exc.printStackTrace();
            throw new FingerprintException(exc);
        }
    }
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class StartActivity extends AppCompatActivity implements
View.OnClickListener {

    Button btnSubmit;
    EditText Username, Password, Price, Route;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_start);

        btnSubmit = (Button) findViewById(R.id.btnSubmit);
        btnSubmit.setOnClickListener(this);

        Username = (EditText) findViewById(R.id.Username);
        Password = (EditText) findViewById(R.id.Password);
        Price = (EditText) findViewById(R.id.Price);
        Route = (EditText) findViewById(R.id.Route);
    }

    @Override
    public void onClick(View v) {
        String userName = Username.getText().toString();
        String password = Password.getText().toString();
        String price = Price.getText().toString();
        String route = Route.getText().toString();
Intent mainActivity = new Intent(this, MainActivity.class);
        mainActivity.putExtra("username", userName);
        mainActivity.putExtra("password", password);
        mainActivity.putExtra("price", price);
        mainActivity.putExtra("route", route);
        startActivity(mainActivity);
    }
}
```

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="1"
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.filters import threshold_otsu
import numpy as np
from glob import glob
from scipy import misc
from matplotlib.patches import Circle,Ellipse
from matplotlib.patches import Rectangle
import os
```

```
from PIL import Image
import keras
from matplotlib import pyplot as plt
import numpy as np
import gzip
%matplotlib inline
from keras.layers import Input,Conv2D,MaxPooling2D,UpSampling2D
from keras.models import Model
from keras.optimizers import RMSprop
from keras.layers.normalization import BatchNormalization

data = glob('FVC2002/Db*/*')
images = []
def read_images(data):
    for i in range(len(data)):
        img = misc.imread(data[i])
        img = misc.imresize(img,(224,224))
        images.append(img)
    return images

images = read_images(data)

images_arr = np.asarray(images)
images_arr = images_arr.astype('float32')

for i in range(2):
    plt.figure(figsize=[5, 5])
    curr_img = np.reshape(images_arr[i], (224,224))
    plt.imshow(curr_img, cmap='gray')
    plt.show()

images_arr = images_arr.reshape(-1, 224,224, 1)
images_arr = images_arr / np.max(images_arr)


from sklearn.model_selection import train_test_split
train_X,valid_X,train_ground,valid_ground = train_test_split(images_arr,
                                                             images_arr,
                                                             test_size=0.2,

random_state=13)
batch_size = 128
epochs = 200
```

```
inChannel = 1
x, y = 224, 224
input_img = Input(shape = (x, y, inChannel))

def autoencoder(input_img):
    conv1 = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_img) #28 x 28 x 32
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1) #14 x 14 x 32
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
#14 x 14 x 64
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2) #7 x 7 x 64
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
#7 x 7 x 128 (small and thick)

    #decoder
    conv4 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
#7 x 7 x 128
    up1 = UpSampling2D((2,2))(conv4) # 14 x 14 x 128
    conv5 = Conv2D(64, (3, 3), activation='relu', padding='same')(up1) # 14
x 14 x 64
    up2 = UpSampling2D((2,2))(conv5) # 28 x 28 x 64
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(up2)
# 28 x 28 x 1
    return decoded


autoencoder = Model(input_img, autoencoder(input_img))
autoencoder.compile(loss='mean_squared_error', optimizer = RMSprop())

autoencoder_train = autoencoder.fit(train_X, train_ground,
batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(valid_X,
valid_ground))

loss = autoencoder_train.history['loss']
val_loss = autoencoder_train.history['val_loss']
epochs = range(200)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```

```
autoencoder = autoencoder.save_weights('autoencoder.h5')
autoencoder = Model(input_img, autoencoder(input_img))
autoencoder.load_weights('autoencoder.h5')
autoencoder.compile(loss='mean_squared_error', optimizer = RMSprop())
```

ДОДАТОК Б

Посібник користувача

ЗМІСТ

В.1 ПІДГОТОВКА ДО РОБОТИ

Для початку роботи із клієнтським додатком необхідний смартфон під управлінням операційної системі «Android» з версією операційної системи не меншій ніж 23 (6.0). Для роботи клієнтського додатка необхідне з’єднання з мережею «інтернет». Користувач клієнтського додатку повинний бути ознайомлений з правилами користування програмою, а також з інструкцією користувача.

В.2 ОПИС ОПЕРАЦІЙ

В.2.1 Перший запуск клієнтського додатку

Для того, щоб успішно авторизуватися компанії перевізнику необхідно, находячись на екрані входу в систему, дані для доступу до сервера (логін та пароль), ввести ціну проїзду та номер маршруту, за яким прямує транспорт. Приклад екрану входу наведено на рисунку В.2.1.

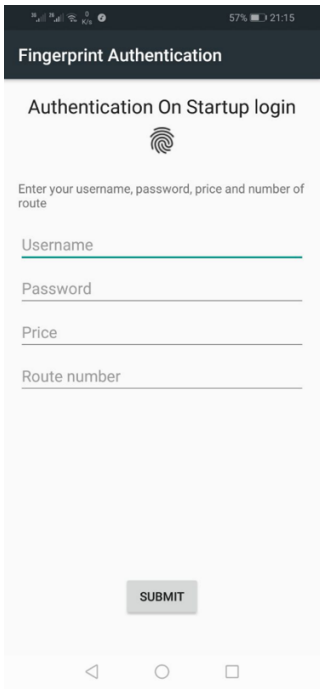

Рисунок В.2.1 – Вид клієнтського додатку після першого входу

На рисунку В.2.2 наведено приклад введених даних. Логін вказано як admin, пароль як adminpw, вартість проїзду як 2, номер маршруту як 134.

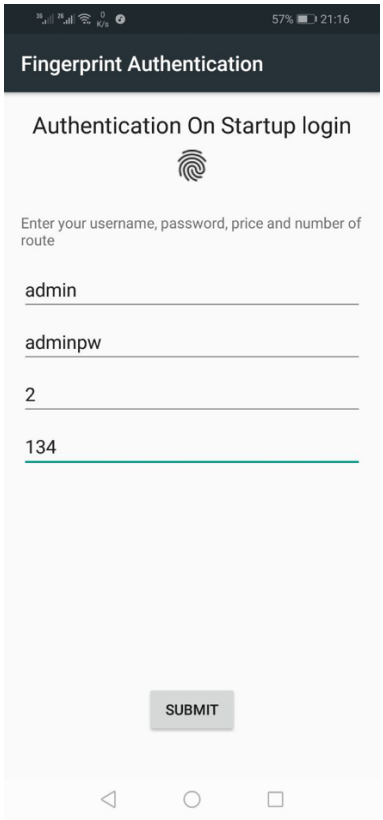

Рисунок В.2.2 – Приклад введений даних після першого входу до клієнтського додатку

Після чого треба натиснути та кнопку «SUBMIT». Після натискання відкриється вікно для сплати транспорту клієнтами компанії перевізника.

В.2.2 Використання додатку

Для того щоб скористатись додатком клієнту потрібно бути зареєстрованим в системі. Для використання системи треба передати дані відбитку (скориставшись сканером відбитків пальців) та дочекатися відповіді від додатку про успішну сплату, чи проблеми, що завадили провести успішну транзакцію. Приклад головного екрану клієнтського додатку наведено на рисунку В.2.3.

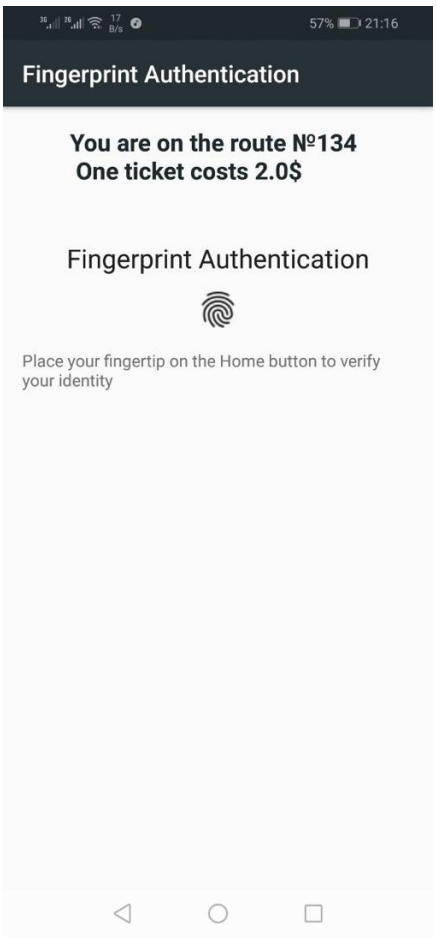

Рисунок В.2.3 – Клієнтський додаток після вводу первинних даних перевізником

В.2.3 Повідомлення про сплату

Під час використання клієнтського додатку, додаток виводить на екран відповідь від сервера про успішність транзакції. Якщо сплата пройшла успішно то додаток виведе «Authorization successful! Transaction successful.». Якщо вийшла помилка при авторизації то додаток виведе «Authentication failed». Приклад таких повідомлень представлено на рисунках В.2.4 та В.2.5.

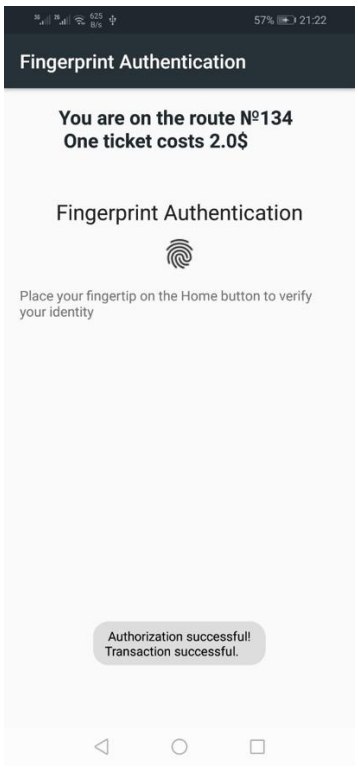

Рисунок В.2.4 – Приклад повідомлення про успішну сплату проїзду

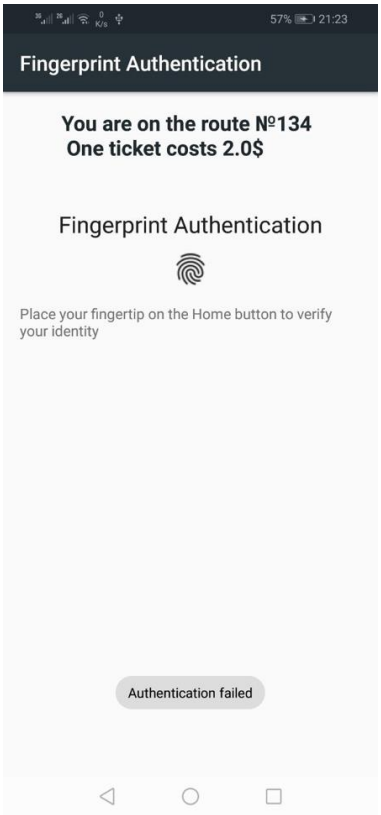

Рисунок В.2.5 – Приклад повідомлення про невдалу сплату проїзду

В.3 АВАРІЙНІ СИТУАЦІЇ

Аварійна ситуація можлива при виході з ладу апаратного забезпечення, яке використовується для роботи клієнтського чи серверного додатку. Для вирішення проблеми необхідно замінити елемент, що викликав причину непрацездатності.

Клієнтський додаток може бути використаним без підключення до мережі інтернет, але без інтернету він стає ефективно даремним бо не може спілкуватися з серверною частиною додатку.

В.4 РЕКОМЕНДАЦІЇ ПО ОСВОЄННЮ

Для успішної взаємодії кожен користувач повинен бути зареєстрованим для користування системою. Специфічні знання в предметній області додатку не потрібні кінцевим користувачам. Компаніям перевізникам необхідно знати правила заповнення даних для автентифікації перевізника при першому старті клієнтської версії додатку.

В.5 КОНТРОЛЬНИЙ ПРИКЛАД

Для контрольного прикладу будуть використовуватися дані, що були введені вище, тобто ціна маршруту буде складати 2.

Приклад інформації про користувача представлено на рисунку В.5.1. Поле «amount» емулює стан рахунку користувача.

Інформація про користувача після сплати представлена на рисунку В.5.2.

Для тесту була взятий користувач під ідентифікатором номер 3.

Рисунок В.5.1 – Приклад інформації про користувача

Рисунок В.5.2 – Приклад інформації про користувача після сплати проїзду

Як ми бачимо після успішної сплати рахунок користувача змінився на вартість проїзду у транспорті

ДОДАТОК В
Специфікація розробленого додатку

Позначення	Найменування	Додаткові відомості
	Документація	
1.	Текст програми	Додаток А
2.	Керівництво користувача	Додаток Б

		Прізвище та ініціали.	Підп.	Дата		
Розробив		Еккер А.В.			Розробка підсистеми оплати міського транспорту на основі біометричних даних	Шифр групи: СШІм-19-1; Код напр./спец.: 122
Перевірив		Золотухін О.В				
Н. Контр.		Малєєва І.А.				ХНУРЕ Кафедра ШІ
Затвердж.		Філатов В.О.				

ДОДАТОК Г

Відомість атестаційної роботи

Відомість атестаційної роботиПозначення	Найменування	Додаткові відомості
	Текстові документи	
1.	Пояснювальна записка	118 с.
	Інші документи	
2.	Презентаційні матеріали	23 плакати

		Прізвище та ініціали.	Підп.	Дата			
Розробив		Еккер А.В.			Розробка підсистеми оплати міського транспорту на основі біометричних даних	Шифр групи	Код напр./спец.
Перевірив		Золотухін О.В				СШІм-19-1	122
Н. Контр.		Малєєва І.А.				ХНУРЕ Кафедра ШІ	
Затвердж.		Філатов В.О.					