MINISTRY OF EDUCATION
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY
COLLEGE OF COMPUTER SCIENCE
& INFORMATION TECHNOLOGY

وزارة التعليم
جامعة الإمام
عبد الرحمن بن فيصل
كلية علوم الحاسب
وتقنية المعلومات

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

CYS 404 – Information Systems Audit

Academic Year 2024 – 2025

Project

Student
Raghad Aljassim
Solaf Mohammed Alhotailah
Aishah Saleh AlKatheer
Abrar AlGhamdi
Remas Alshammari

Table of Contents

List of Tables:

List of Figures:

1. Executive summary:

This report shares our journey auditing the cybersecurity of a Restaurant Management System we built during our OOP2 course using Java and MySQL. The system handles essential restaurant tasks like taking orders, managing customer data, billing, and keeping track of inventory so making sure it's secure felt incredibly important.

Our goal was to see how well the system aligns with the National Cybersecurity Authority’s Essential Cybersecurity Controls (ECC v2.0) by focusing on nine key subdomains. Throughout the audit, we discovered some serious gaps especially in areas like risk management, access controls, and protecting sensitive data.

Instead of just pointing out problems, we worked on realistic and budget-friendly solutions to help strengthen the system’s security. This experience wasn’t just about ticking boxes, it helped us connect everything we have learned about cybersecurity to a real project we were involved in. It made the importance of secure systems feel more real, and it taught us a lot about what it takes to build systems that are not just functional, but safe too.

2. Methodology followed during the audit:

To carry out our audit, we followed a clear and practical seven-step process that helped us stay organized and focused:

1. **Choosing the System:**
   We picked a system we knew inside and out our own Restaurant Management System. Since we built it ourselves, we have had full access to everything: the source code, database, and user logic. That made it the perfect candidate for a detailed security check.

2. **Selecting ECC Subdomains:**
   From the full ECC v2.0 framework, we carefully selected 9 subdomains that made the most sense for our system things like data protection, access management, and backups.

3. **Mapping Controls to Our System:**
   We took each ECC control and compared it with what our system currently does. This helped us see where things lined up and where they didn't.

4. **Collecting Evidence:**
   We dove into the backend examining Java code, SQL queries, database tables, user roles, and how authentication was handled. We also took notes on how the system responded in different scenarios.

5. **Finding the Gaps:**
   Once we had the full picture, we checked which controls were being followed (✅) and which weren't (❌). This helped us highlight the weak spots.

6. **Identifying Risks:**
   We discussed what could go wrong like SQL injection, data leaks, or unauthorized access and documented all potential threats for each area.

7. **Making Realistic Recommendations:**
   Finally, we brainstormed and suggested practical steps to improve security. Our focus was on simple, achievable changes that would make a big difference.

Tools We Used:

- NetBeans IDE – to explore and test the Java source code
- MySQL Workbench – to review and interact with the database
- Team Meetings – to review findings, clarify code logic, and interview each other for insights

3. Selected ECC Subdomains Assessment:

3.1. Subdomain 1-5: Cybersecurity Risk Management

a. **Subdomain Title:** Cybersecurity Risk Management

b. **Description of Relevance:** Effective risk management is vital for the restaurant management system to proactively identify and mitigate threats that could compromise customer data, inventory management, and operational integrity. This ensures that risks are managed according to industry standards, protecting both the business and its customers.

c. **Assessment Table:**

Control	Description	Current Practice	Compliant (✅ / ❌)	Notes / Gaps / Issues Identified
1-5-1	Define a risk management methodology for the system.	No risk strategy or documentation.	❌	No methodology for identifying, documenting, or responding to system risks.
1-5-2	Conduct regular risk assessments focused on system vulnerabilities.	No vulnerability scans or reviews.	❌	No security testing or vulnerability scanning in place (e.g., input validation, SQL injection).
1-5-3-1	Assess risks during the initial stages of system updates.	Updates made without review.	❌	Code changes are applied directly with no prior risk checks.
1-5-3-2	Evaluate risks before major changes to infrastructure.	Direct DB changes; no planning.	❌	Database passwords are hardcoded; infrastructure changes are untracked.
1-5-3-3	Assess risks when engaging third-party services.	No third-party services used.	❌	Currently not applicable.
1-5-3-4	Evaluate risks before launching new features.	Features added without review.	❌	New GUI features are implemented without testing or risk analysis.
1-5-4	Periodically review the risk management process.	No reviews or updates to process.	❌	No regular process evaluation or updates performed.

Table 1: Assessment Table of Subdomain 1-5: Cybersecurity Risk Management.

d. Recommendations:

1. Implement a Basic Security Risk Checklist

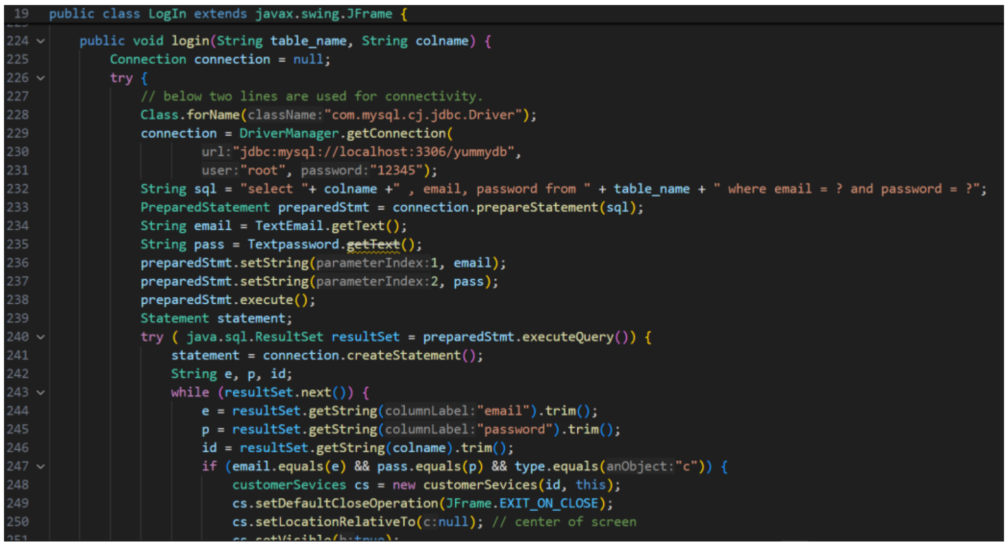
- Implement a concise security checklist (e.g., input validation, access controls, credential management) to be completed before code commits or feature deployments.
- Integrate this checklist into existing workflows (e.g., Git hooks, pull request templates) to minimize disruption.

2. Adopt Automate Vulnerability Detection in Development

- Deploy static application security testing (SAST) tools (e.g., SonarQube, SpotBugs) to identify critical vulnerabilities (e.g., hardcoded credentials, injection risks) during development.
- Prioritize addressing high-severity findings to maintain focus on material risks.

3. Maintain Documentation for Risk Tracking

- Establish a living document (e.g., internal wiki, changelog) to track security-relevant changes and mitigation measures.
- Reference standardized frameworks (e.g., OWASP Top 10) to ensure consistent risk assessment and response protocols.



```java
public class LogIn extends javax.swing.JFrame {
    public void login(String table_name, String colname) {
        Connection connection = null;
        try {
            // below two lines are used for connectivity.
            Class.forName(className:"com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(
                    url:"jdbc:mysql://localhost:3306/yummydb",
                    user:"root", password:"12345");
            String sql = "select "+ colname +" , email, password from " + table_name + " where email = ? and password = ?";
            PreparedStatement preparedStmt = connection.prepareStatement(sql);
            String email = TextEmail.getText();
            String pass = Textpassword.getText();
            preparedStmt.setString(parameterIndex:1, email);
            preparedStmt.setString(parameterIndex:2, pass);
            preparedStmt.execute();
            Statement statement;
            try ( java.sql.ResultSet resultSet = preparedStmt.executeQuery()) {
                statement = connection.createStatement();
                String e, p, id;
                while (resultSet.next()) {
                    e = resultSet.getString(columnLabel:"email").trim();
                    p = resultSet.getString(columnLabel:"password").trim();
                    id = resultSet.getString(colname).trim();
                    if (email.equals(e) && pass.equals(p) && type.equals(anObject:"c")) {
                        customerSevices cs = new customerSevices(id, this);
                        cs.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        cs.setLocationRelativeTo(c:null); // center of screen
```

Figure 1: 1-5: The hardcoded credentials represent a significant risk. A risk management methodology should identify this as a critical vulnerability. Regular risk assessments should detect this flaw.

3.2. Subdomain 2-1: Asset Management.

a. **Subdomain Title:** Asset Management.

b. **Description of Relevance:** Asset management is critical for tracking and protecting the information and technology employed in the restaurant management system. This guarantees the safety, compliance, and efficient management of every element, including servers, software, and customer information.

c. **Assessment Table:**

Control	Description	Current Practice	Compliant (✅ / ❌)	Notes / Gaps / Issues Identified
2-1-1	Document asset requirements for all system components.	Uses Java classes to define data fields (ID, name, contact info).	✅	Basic asset tracking exists (Admin/Customer classes), but no formal documentation.
2-1-2	Implement processes to track hardware and software assets.	Tracks user accounts (admins/customers) but not devices or software licenses.	❌	No system to track hardware or software licenses.
2-1-3	Maintain updated records of all assets.	Updates customer/admin records via setters (e.g., setPhoneNumber().	✅	Getters/setters in Admin and Customer classes allow updates, but no audit logs.
2-1-4	Conduct regular reviews of asset inventory.	There are no regular reviews only ad hoc checks are performed when an issue arises.	❌	No automated or scheduled review process.
2-1-5	Tag and track assets throughout their lifecycle.	Uses database IDs for customer but no lifecycle tags.	❌	No unique identifiers or lifecycle states (e.g., "active," "undergoing repair").
2-1-6	Periodically review asset management practices.	There are no official reviews, everything is up to the developer's discretion.	❌	No policy or process for periodic reviews.

Table 2: Assessment Table of Subdomain 2-1: Asset Management.

d. Recommendations:

1. Centralized Asset Inventory System.

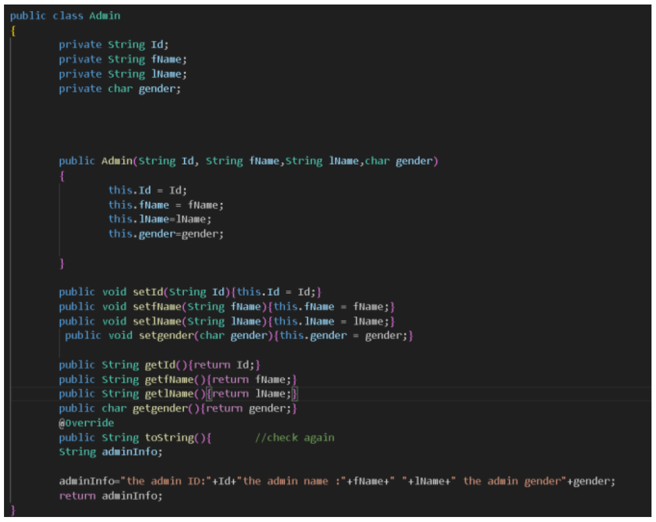
- Create a centralized inventory system for all hardware and software licenses used in the restaurant management system.

2. Routine Asset Record Audits.

- Perform routine audits to confirm asset records' accuracy and make sure internal policies are being followed.

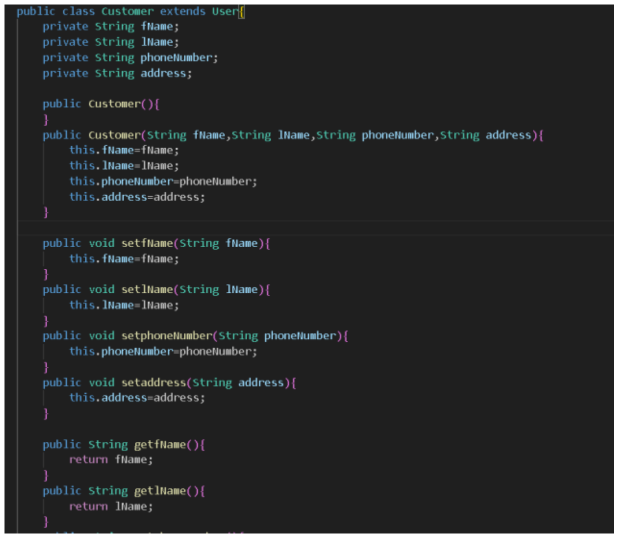
3. Comprehensive Asset Change Logging.

- Keep complete change logs for all asset modifications, including who authorized updates, when they occurred, and why changes were made, to promote accountability.



```
public class Admin
{
        private String Id;
        private String fName;
        private String lName;
        private char gender;



        public Admin(String Id, String fName,String lName,char gender)
        {
                this.Id = Id;
                this.fName = fName;
                this.lName=lName;
                this.gender=gender;

        }

        public void setId(String Id){this.Id = Id;}
        public void setfName(String fName){this.fName = fName;}
        public void setlName(String lName){this.lName = lName;}
         public void setgender(char gender){this.gender = gender;}

        public String getId(){return Id;}
        public String getfName(){return fName;}
        public String getlName(){return lName;}
        public char getgender(){return gender;}
        @Override
        public String toString(){       //check again
        String adminInfo;

        adminInfo="the admin ID:"+Id+"the admin name :"+fName+" "+lName+" the admin gender"+gender;
        return adminInfo;
}
```

Figure 2: 2-1-1: Document asset requirements for all system components.



```
public class Customer extends User{
    private String fName;
    private String lName;
    private String phoneNumber;
    private String address;

    public Customer(){
    }
    public Customer(String fName,String lName,String phoneNumber,String address){
        this.fName=fName;
        this.lName=lName;
        this.phoneNumber=phoneNumber;
        this.address=address;
    }

    public void setfName(String fName){
        this.fName=fName;
    }
    public void setlName(String lName){
        this.lName=lName;
    }
    public void setphoneNumber(String phoneNumber){
        this.phoneNumber=phoneNumber;
    }
    public void setaddress(String address){
        this.address=address;
    }

    public String getfName(){
        return fName;
    }
    public String getlName(){
        return lName;
    }
```

Figure 3: 2-1-3: Maintain updated records of all assets.

3.3. Subdomain 2-2: Identity and Access Management.

a. **Subdomain Title:** Identity and Access Management.

b. **Description of Relevance:** This subdomain plays a critical role in ensuring that information and technological assets are logically accessible and safe, preventing unwanted access and granting only authorized access to users who are required to complete tasks.

c. **Assessment Table:**

Control	**Description**	**Current Practice**	**Compliant (✅ / ❌)**	**Notes / Gaps / Issues Identified**
2-2-1	Define, document, and provide approval for identity and access management flows.	There is no official documentation or policy in place.	❌	Although access control is directly coded, neither police nor established standards support it.
2-2-2	Set up and execute requirements for identity and access management processes.	Before using any features, users must log in.	✅	There is a basic implementation using a login system.
2-2-3-1	A combination of username and password for credential verification.	A username and password are needed to log in.	✅	Passwords need to be hashed as they are saved in plain text.
2-2-3-2	Securing remote access via Multi-Factor Authentication (MFA).	There is no second layer of authentication.	❌	Not implemented.
2-2-3-3	Authorization based on Need-to Know and Need-to-Use, Least Privilege and Segregation of Duties.	Customers and administrators have different access and roles.	✅	There is functional role segregation, but no fine-grained permission model is in place.

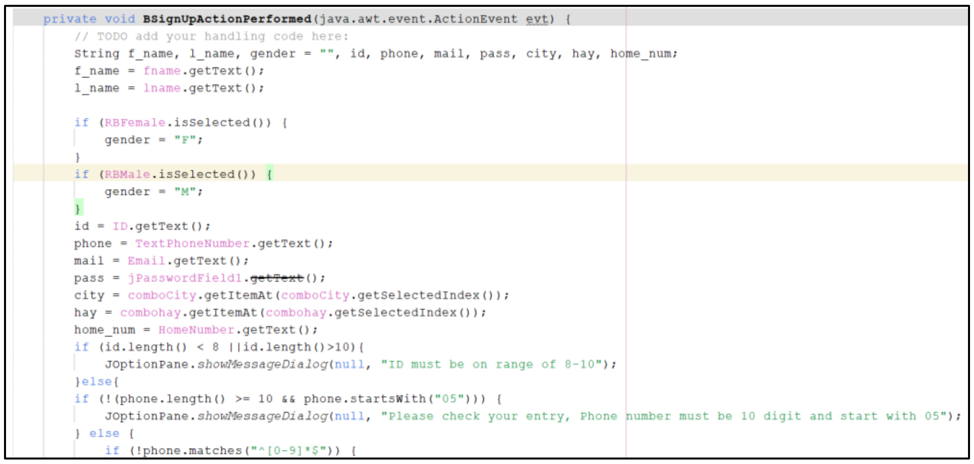
Table 3: Assessment Table of Subdomain 2-2: Identity and Access Management. 1/2

Control	Description	Current Practice	Compliant (✅ / ❌)	Notes / Gaps / Issues Identified
2-2-3-4	Privileged access management.	Although administrators have more access, privileged accounts are not further protected or have logging.	⚠	There are no safeguards for enhanced privileges or audit trails.
2-2-3-5	Periodic review of identities and access rights.	No procedure for review or expiration.	❌	Access cannot be reviewed, deactivated, or roles reassigned once it has been given.
2-2-4	The Implementation of the cybersecurity requirements for identity and access management must be reviewed periodically.	The access control model has not been reviewed or updated.	❌	The IAM system is static and does not undergo regular evaluation or modification to account for shifts in user roles.

Table 4: Assessment Table of Subdomain 2-2: Identity and Access Management. 2/2

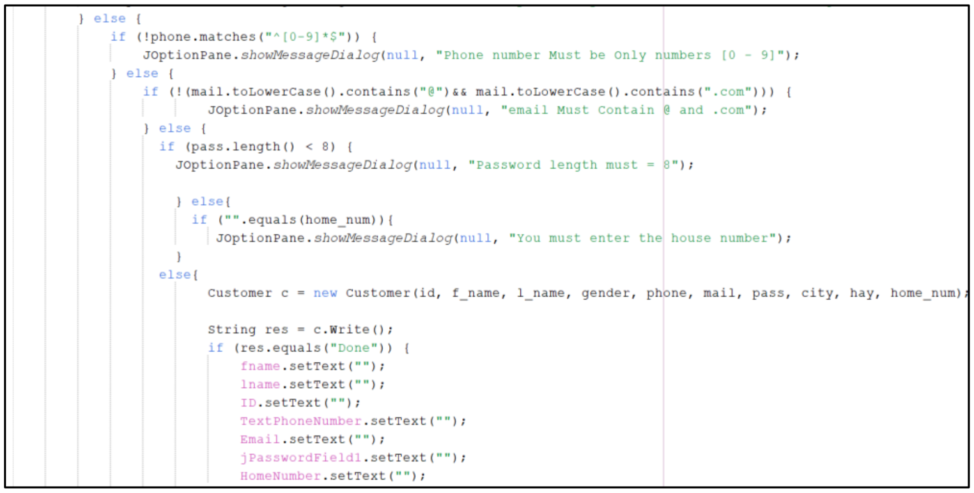
d. Recommendations:

- Employ password hashing to secure sensitive information such as passwords.
- Require Multi-Factor Authentication (MFA) on administrator accounts.
- Enforce Role-Based Access Control (RBAC), ensuring all roles are clearly defined and permissions are outlined.
- Track admin activity for auditing and accountability purposes.
- Configure sessions to time out after periods of inactivity, automatically logging out inactive users.
- Review access permissions periodically to ensure accuracy and relevance, removing any outdated permissions.
- Implement IAM policies and procedures within documented frameworks.
- Guard against injection attacks by validating user input on login and registration forms.



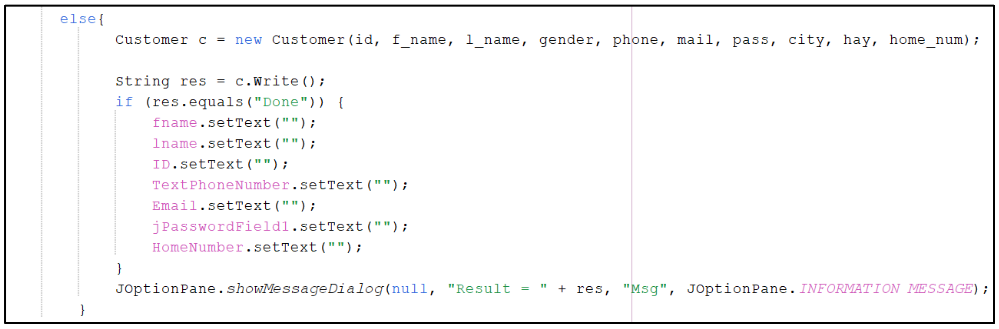
```
private void BSignUpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String f_name, l_name, gender = "", id, phone, mail, pass, city, hay, home_num;
    f_name = fname.getText();
    l_name = lname.getText();

    if (RBFemale.isSelected()) {
        gender = "F";
    }
    if (RBMale.isSelected()) {
        gender = "M";
    }
    id = ID.getText();
    phone = TextPhoneNumber.getText();
    mail = Email.getText();
    pass = jPasswordField1.~~getText~~();
    city = comboCity.getItemAt(comboCity.getSelectedIndex());
    hay = combohay.getItemAt(combohay.getSelectedIndex());
    home_num = HomeNumber.getText();
    if (id.length() < 8 ||id.length()>10){
        JOptionPane.showMessageDialog(null, "ID must be on range of 8-10");
    }else{
    if (!(phone.length() >= 10 && phone.startsWith("05"))) {
        JOptionPane.showMessageDialog(null, "Please check your entry, Phone number must be 10 digit and start with 05");
    } else {
        if (!phone.matches("^[0-9]*$")) {
```

Figure 4: 2-2-2: Identity and Access Management.



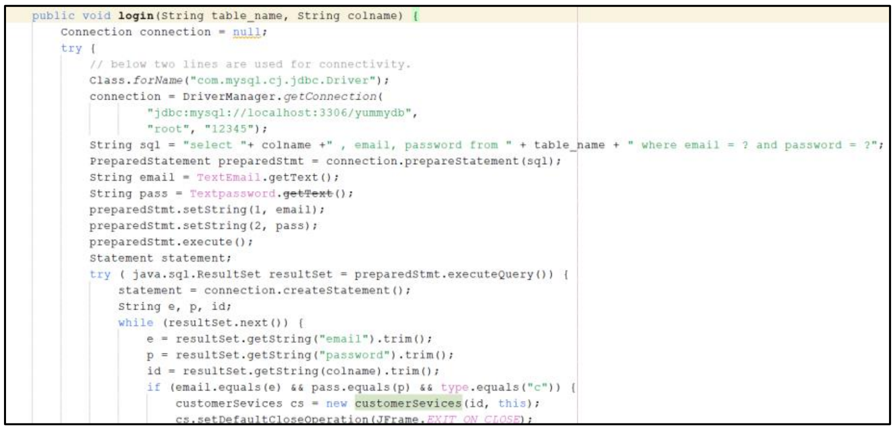
```
    } else {
        if (!phone.matches("^[0-9]*$")) {
            JOptionPane.showMessageDialog(null, "Phone number Must be Only numbers [0 - 9]");
        } else {
            if (!(mail.toLowerCase().contains("@")&& mail.toLowerCase().contains(".com"))) {
                    JOptionPane.showMessageDialog(null, "email Must Contain @ and .com");
            } else {
              if (pass.length() < 8) {
                JOptionPane.showMessageDialog(null, "Password length must = 8");

                } else{
                  if ("".equals(home_num)){
                   JOptionPane.showMessageDialog(null, "You must enter the house number");
                }
              else{
                    Customer c = new Customer(id, f_name, l_name, gender, phone, mail, pass, city, hay, home_num);

                    String res = c.Write();
                    if (res.equals("Done")) {
                        fname.setText("");
                        lname.setText("");
                        ID.setText("");
                        TextPhoneNumber.setText("");
                        Email.setText("");
                        jPasswordField1.setText("");
                        HomeNumber.setText("");
```

Figure 5: 2-2-2: Identity and Access Management.



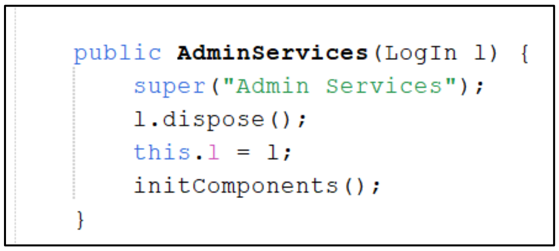
```
  else{
        Customer c = new Customer(id, f_name, l_name, gender, phone, mail, pass, city, hay, home_num);

        String res = c.Write();
        if (res.equals("Done")) {
            fname.setText("");
            lname.setText("");
            ID.setText("");
            TextPhoneNumber.setText("");
            Email.setText("");
            jPasswordField1.setText("");
            HomeNumber.setText("");
        }
        JOptionPane.showMessageDialog(null, "Result = " + res, "Msg", JOptionPane.INFORMATION_MESSAGE);
  }
```

Figure 6: 2-2-2: Identity and Access Management.



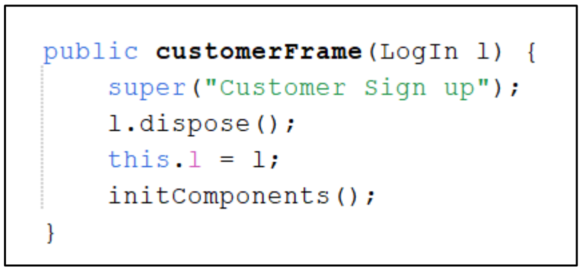
```
public void login(String table_name, String colname) {
    Connection connection = null;
    try {
        // below two lines are used for connectivity.
        Class.forName("com.mysql.cj.jdbc.Driver");
        connection = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/yummydb",
                "root", "12345");
        String sql = "select "+ colname +" , email, password from " + table_name + " where email = ? and password = ?";
        PreparedStatement preparedStmt = connection.prepareStatement(sql);
        String email = TextEmail.getText();
        String pass = Textpassword.~~getText~~();
        preparedStmt.setString(1, email);
        preparedStmt.setString(2, pass);
        preparedStmt.execute();
        Statement statement;
        try ( java.sql.ResultSet resultSet = preparedStmt.executeQuery()) {
            statement = connection.createStatement();
            String e, p, id;
            while (resultSet.next()) {
                e = resultSet.getString("email").trim();
                p = resultSet.getString("password").trim();
                id = resultSet.getString(colname).trim();
                if (email.equals(e) && pass.equals(p) && type.equals("c")) {
                    customerSevices cs = new customerSevices(id, this);
                    cs.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Figure 7: 2-2-3-1: Identity and Access Management.



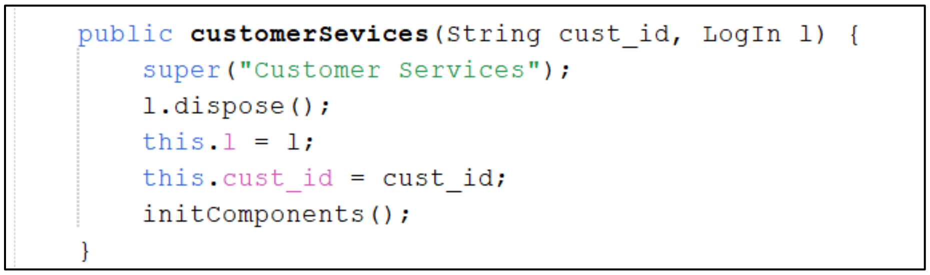
```
public AdminServices(LogIn l) {
    super("Admin Services");
    l.dispose();
    this.l = l;
    initComponents();
}
```

Figure 8: 2-2-3-3: Identity and Access Management.



```
public customerFrame(LogIn l) {
    super("Customer Sign up");
    l.dispose();
    this.l = l;
    initComponents();
}
```

Figure 9: 2-2-3-3: Identity and Access Management.



```
public customerSevices(String cust_id, LogIn l) {
    super("Customer Services");
    l.dispose();
    this.l = l;
    this.cust_id = cust_id;
    initComponents();
}
```

Figure 10: 2-2-3-3: Identity and Access Management.

3.4. Subdomain 2-9: Backup and Recovery Management.

a. **Subdomain Title:** Backup and Recovery Management.

b. **Description of Relevance:** Setting up trustworthy backup and restoration systems is critical for safeguarding data against loss resulting from cyberattacks, hardware malfunction, or human mistakes. Doing so enables the organization to swiftly resume its activities, curtailing downtime and maintaining both data correctness and service functionality.

c. **Assessment Table:**

Control	Description	Current Practice	Compliant (✅ / ❌)	Notes / Gaps / Issues Identified
2.9.1	Automated backups should be executed on a recurring basis	A dedicated backup program handles data preservation tasks, performing these actions on a daily schedule, without any manual intervention	❌	Occasional failures not always detected
2.9.2	Data requiring preservation should undergo encryption, both while stored and during transmission	Data is encrypted using AES-256 during backup	❌	data at rest and in transmission outside of backups may not be covered
2.9.3	Data backups necessitate being kept at locales physically distant from the primary site	Backups stored only on local server	❌	No offsite/cloud backup currently used
2.9.4	Documenting and testing procedures for backup recovery is essential	Basic documentation exists, testing done once last year	❌	No scheduled testing routine
2.9.5	Securing backup availability is crucial authorized personnel only	IAM roles enforced on backup systems	❌	gaps exist in periodic review or least privilege enforcement
2.9.6	Data retention policies must be clearly defined and followed	Retention periods not standardized	❌	No documented policy or automation

Table 5: Assessment Table of Subdomain 2-9: Backup and Recovery Management.

d. Recommendations:

1. Establish Real-Time Monitoring and Alerting for Backup Operations.

- Integrate notifications from your backup software (if it offers such a feature), cron job email reports, or deploy third-party monitoring solutions. The aim is to instantly inform IT personnel of any backup failures or instances where a backup process doesn't finish as intended.

2. Establish Remote or Cloud-Hosted Backup.

- Employ cloud platforms such as Amazon S3, Google Cloud Storage, or Azure Backup to ensure safe, offsite data storage. This strategy shields data from the impact of local catastrophes.

3. Develop and Quarterly Evaluate a Recovery Blueprint.

- Establish detailed, procedural instructions for data restoration, then undertake simulated recovery exercises (drills) on a three-monthly basis. This will ensure team competency in expediently restoring data during an actual disruptive event.

4. Establish and Automate Data Retention Rules Driven by Data Categories.

- Implement backup management systems that apply varied retention schedules (for instance, 30 days for log files, and 12 months for student data). This ensures that data handling remains both consistent and compliant with the required standards.

5. Control Backup Access with Role-Based Security.

- Grant backup access solely to designated backup administrators and system engineers. Implement Multi-Factor Authentication (MFA) for all access attempts and meticulously record every activity related to access.

3.5. Subdomain 2-10: Vulnerability Management.

a. **Subdomain Title:** Vulnerability Management

b. **Description of Relevance:** To guard against potential cyberthreats and maintain operational continuity, the restaurant management system must routinely discover and fix vulnerabilities. This preserves trust and protects consumer data.

c. **Assessment Table:**

Control	**Description**	**Current Practice**	**Compliant (✅ / ❌)**	**Notes / Gaps / Issues Identified**
2-10-1	Define a vulnerability management policy.	There is no formal policy, developers fix bugs as reported by users.	❌	No documented process for handling vulnerabilities.
2-10-2	Implement effective vulnerability management processes.	Manual code reviews during development and there are not any automated tools.	❌	No scanning tools or manual review steps.
2-10-3-1	Conduct regular vulnerability assessments.	Tests only during major releases (new menu items).	❌	No automated scans.
2-10-3-2	Classify vulnerabilities by criticality.	All bugs treated equally; no severity tiers ("Critical"/ "Low").	❌	Critical risks unaddressed.
2-10-3-3	Remediate vulnerabilities based on classification.	Fixes are only applied to significant failures like payment failures.	❌	No process for patching.
2-10-3-4	Establish security patch management practices.	Updates third-party libraries manually.	❌	No process to update dependencies.
2-10-3-5	Subscribe to trusted cybersecurity resources.	No subscriptions to CVE databases (e.g., NVD) or threat feeds.	❌	No subscriptions to CVE databases.
2-10-4	Review vulnerability management practices periodically.	No retrospective reviews after incidents.	❌	No periodic vulnerability reviews.

Table 6: Assessment Table of Subdomain 2-10: Vulnerability Management.

d. Recommendations:

1. Automated Vulnerability Scanning Implementation

- Use a vulnerability scanning tool to find vulnerabilities on a regular basis and fix them quickly.

2. Structured Vulnerability Remediation Plan

- In order to lower risks, create a remediation strategy with specific dates for fixing vulnerabilities found.

3. Scheduled Third-Party Dependency Assessments

- Plan weekly automated vulnerability assessments to find security flaws in third-party dependencies.

3.6. Subdomain 2-13: Cybersecurity Incident and Threat Management.

a. **Subdomain Title:** Cybersecurity Incident and Threat Management.

b. **Description of Relevance:** Cybersecurity incidents and the threats they present demand effective handling; this is fundamental to preserving the confidentiality, integrity, and operational status of organizational infrastructures. This particular area of focus centers around anticipatory identification of risks, rapid reactions to incidents as they unfold, and the ongoing refinement of our defenses to lessen the negative impacts and prevent future occurrences.

c. **Assessment Table:**

Control	Description	Current Practice	Compliant (✅ / ❌)	Notes / Gaps / Issues Identified
2-13-1	Define an incident response policy.	A draft plan is created but not yet approved	❌	No official approval or formal implementation
2-13-2	Establish incident reporting and escalation procedures.	SIEM system deployed and operational	✅	Monitoring is active and alerts generated
2-13-3-1	Document incident handling procedures.	Logs are collected, but reviews are irregular	❌	No fixed review schedule
2-13-3-2	Classify incidents for appropriate response.	Manual recording in spreadsheets	❌	No classification framework
2-13-3-3	Define reporting mechanisms to authorities.	One-time awareness session conducted	❌	No regular or ongoing training
2-13-3-4	Establish procedures for sharing threat intelligence.	No threat intelligence mechanism in place	❌	Lack of threat feed subscriptions or sharing
2-13-3-5	Collect and handle threat intelligence feeds.	One incident review done this year	❌	Not practiced consistently
2-13-4	Review incident management processes periodically.	Standard Operating Procedure (SOP) is in place	✅	Roles and contacts well defined

Table 7: Assessment Table of Subdomain 2-13: Cybersecurity Incident and Threat Management.

d. Recommendations:

1. Establish a Routine for Log Examination.

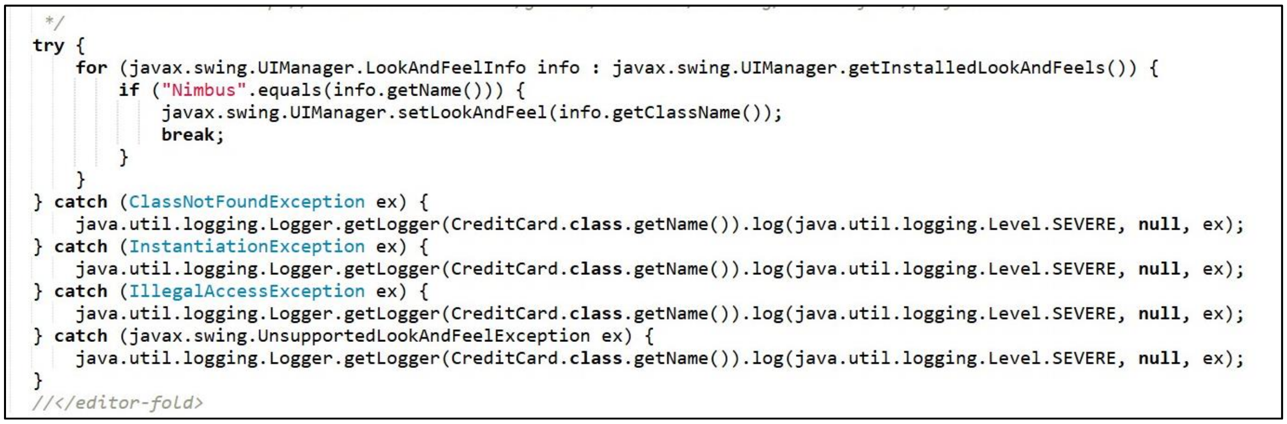
- Designate a particular IT or security professional to conduct a weekly investigation of incident logs. Utilize automated reporting mechanisms to simplify this routine and bring unusual events to immediate attention

2. Implement Cybersecurity Awareness Training.

- Establish a regular training schedule, ensuring sessions occur every six months.

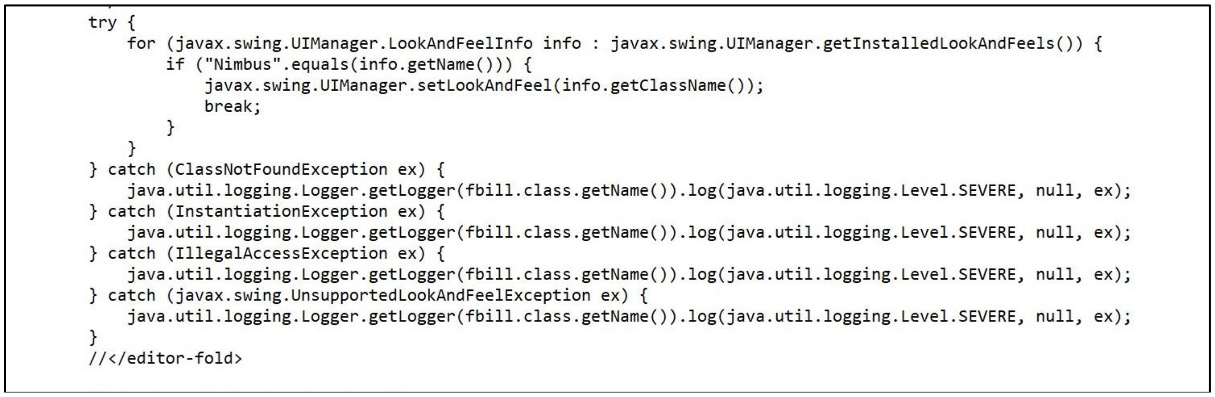
3. Ensure Consistent Post-Incident Assessments.

- After any security event, it is crucial to conduct a detailed investigation, documented comprehensively. Utilize a consistent reporting format. This report should encompass critical aspects: the originating cause, the remedial actions carried out, the principal lessons learned, and recommendations for modifying existing security protocols if warranted.



```
 */
try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(CreditCard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(CreditCard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(CreditCard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(CreditCard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>
```

Figure 11: 2-13-2: Establish incident reporting and escalation procedures.



```
try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(fbill.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(fbill.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(fbill.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(fbill.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>
```

Figure 12: 2-13-4: Review incident management processes periodically.

4. Recommendations

4.1. Cybersecurity Roles:

- Everyone on the team should know their role when it comes to security.
- Just writing down who's responsible for what can make a big difference.

4.2. Risk Management:

- We need to start tracking possible risks—even a simple list is helpful.
- Checking on those risks regularly will help us stay ahead.

4.3. Asset Management:

- We should keep a record of all our system parts, like the database, code, and files.
- It also helps to label which ones are more sensitive.

4.4. Access Control:

- Admin-only features should be protected with proper roles.
- Everyone should have their own login, and admins should use MFA.
- Reviewing logs every now and then can catch anything unusual.

4.5. Data Protection:

- Any personal or important data should be encrypted.
- If we don't need it, we should delete it safely.
- Only those who really need access should have it.

4.6. Cryptography:

- We need to follow standard encryption practices (like bcrypt for passwords).
- Also, keep track of how keys are stored and rotated.

4.7. Backup & Recovery:

- Daily automatic backups will save us if something goes wrong.
- We should also test restoring backups once in a while to make sure it works.

4.8. Vulnerability Management:

- Let’s scan our code with tools to catch bugs or weaknesses.
- And keep a list of what we find so we don’t lose track.

4.9. Incident Response:

- It is smart to have a plan if something bad happens—who does what, and when.
- We should turn on system logs and watch for any strange behavior.