Lab 4 –Fixing Payment Card Industry Java Web Application

Terrel Blackston

Course: SDEV 425

Section: 6980

Professor Ronald McFarland

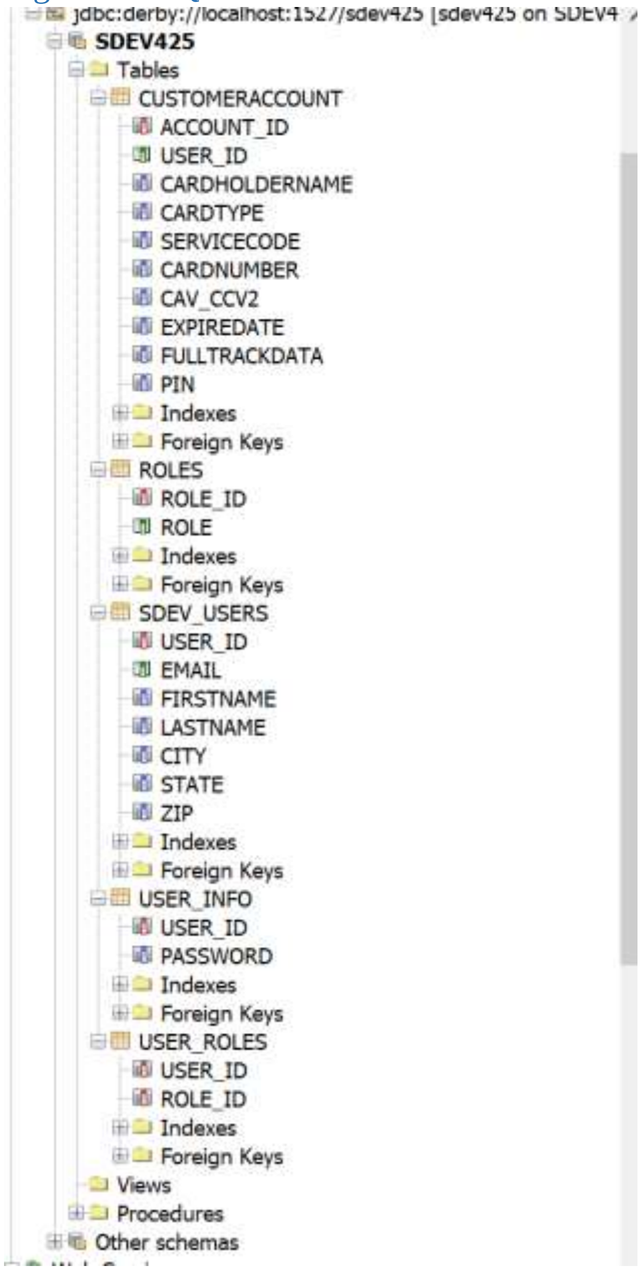
07/28/19

Table of Contents

Part 1: Sample Java Application

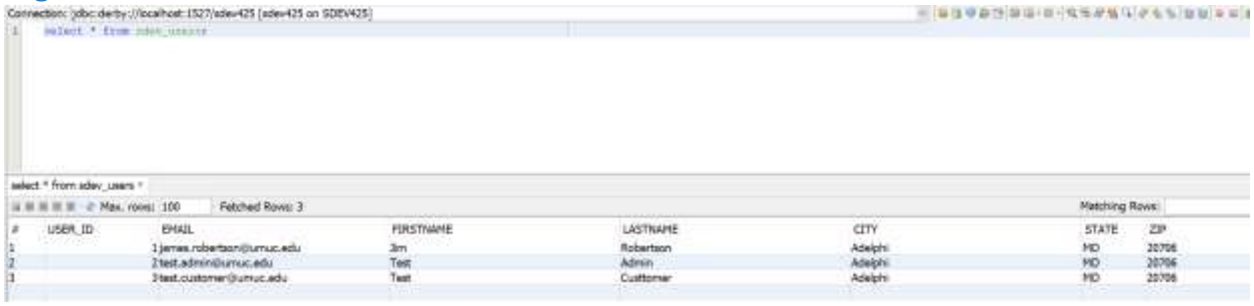
I’ve provided screenshots below that shows the Payment applications working properly, along with pulling data from the server as well.

Figure 1.0 SQL Database and Server:



```
jdbc:derby://localhost:1527/sdev425 [sdev425 on SDEV4
  SDEV425
    Tables
      CUSTOMERACCOUNT
        ACCOUNT_ID
        USER_ID
        CARDHOLDERNAME
        CARDTYPE
        SERVICECODE
        CARDNUMBER
        CAV_CCV2
        EXPIREDATE
        FULLTRACKDATA
        PIN
        Indexes
        Foreign Keys
      ROLES
        ROLE_ID
        ROLE
        Indexes
        Foreign Keys
      SDEV_USERS
        USER_ID
        EMAIL
        FIRSTNAME
        LASTNAME
        CITY
        STATE
        ZIP
        Indexes
        Foreign Keys
      USER_INFO
        USER_ID
        PASSWORD
        Indexes
        Foreign Keys
      USER_ROLES
        USER_ID
        ROLE_ID
        Indexes
        Foreign Keys
    Views
    Procedures
  Other schemas
```

Figure 1.1 Database tables view



```
Connection: jdbc:derby://localhost:1527/sdev425 [sdev425 on SDEV425]
1   select * from sdev_users
```

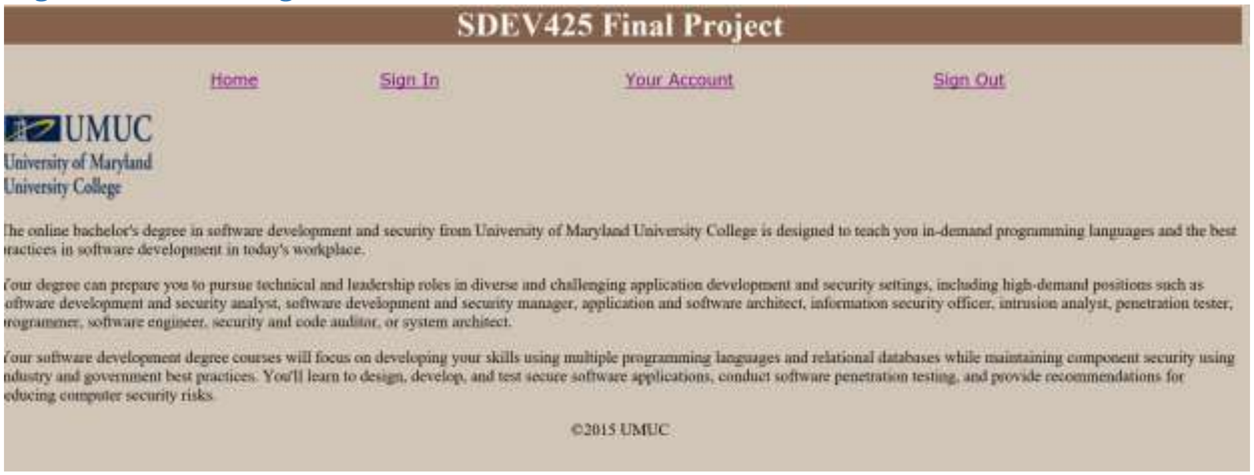
select * from sdev_users

Max. rows: 100 Fetched Rows: 3 Matching Rows:

#	USER_ID	EMAIL	FIRSTNAME	LASTNAME	CITY	STATE	ZIP
1	1	james.robertson@umuc.edu	Jim	Robertson	Adelphi	MD	20706
2	2	test.admin@umuc.edu	Test	Admin	Adelphi	MD	20706
3	3	test.customer@umuc.edu	Test	Customer	Adelphi	MD	20706

Now that we know the data is able to be accessed through the execution commands, I have provide below screenshots of the application in running.

Figure 1.2 Index Page:



SDEV425 Final Project

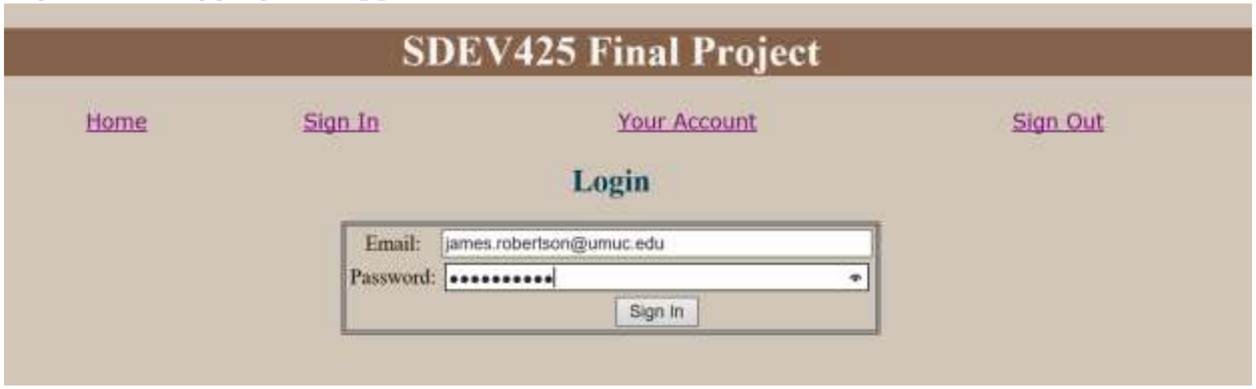
The online bachelor's degree in software development and security from University of Maryland University College is designed to teach you in-demand programming languages and the best practices in software development in today's workplace.

Your degree can prepare you to pursue technical and leadership roles in diverse and challenging application development and security settings, including high-demand positions such as software development and security analyst, software development and security manager, application and software architect, information security officer, intrusion analyst, penetration tester, programmer, software engineer, security and code auditor, or system architect.

Your software development degree courses will focus on developing your skills using multiple programming languages and relational databases while maintaining component security using industry and government best practices. You'll learn to design, develop, and test secure software applications, conduct software penetration testing, and provide recommendations for reducing computer security risks.

©2015 UMUC

Figure 1.3 Logging into application

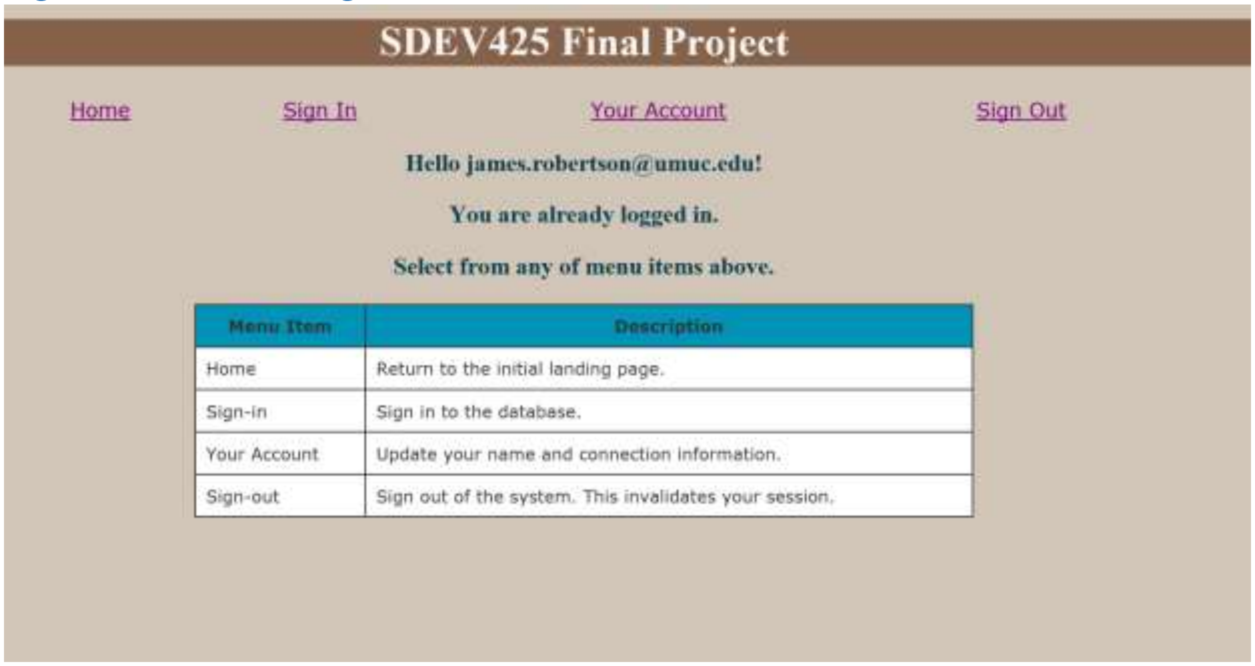


SDEV425 Final Project

Login

Email: james.robertson@umuc.edu
Password: ••••••••••

Figure 1.4 Successful login



SDEV425 Final Project

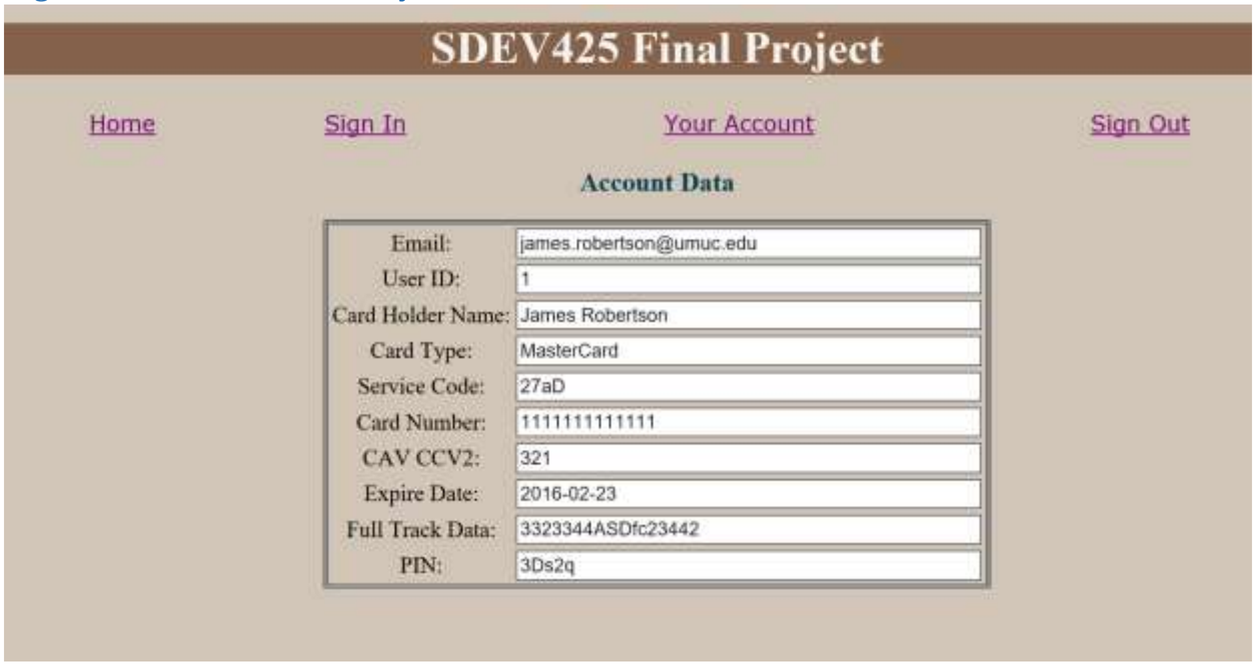
Hello james.robertson@umuc.edu!

You are already logged in.

Select from any of menu items above.

Menu Item	Description
Home	Return to the initial landing page.
Sign-in	Sign in to the database.
Your Account	Update your name and connection information.
Sign-out	Sign out of the system. This invalidates your session.

Figure 1.5 Account Summary

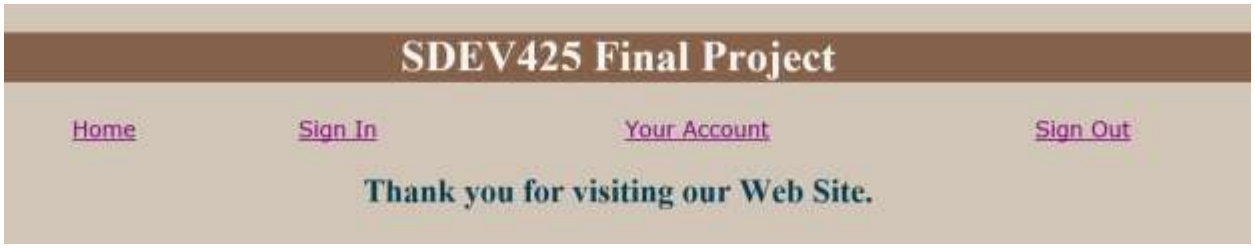


SDEV425 Final Project

Account Data

Email:	james.robertson@umuc.edu
User ID:	1
Card Holder Name:	James Robertson
Card Type:	MasterCard
Service Code:	27aD
Card Number:	1111111111111
CAV CCV2:	321
Expire Date:	2016-02-23
Full Track Data:	3323344ASDfc23442
PIN:	3Ds2q

Figure 1.6 Signing Out

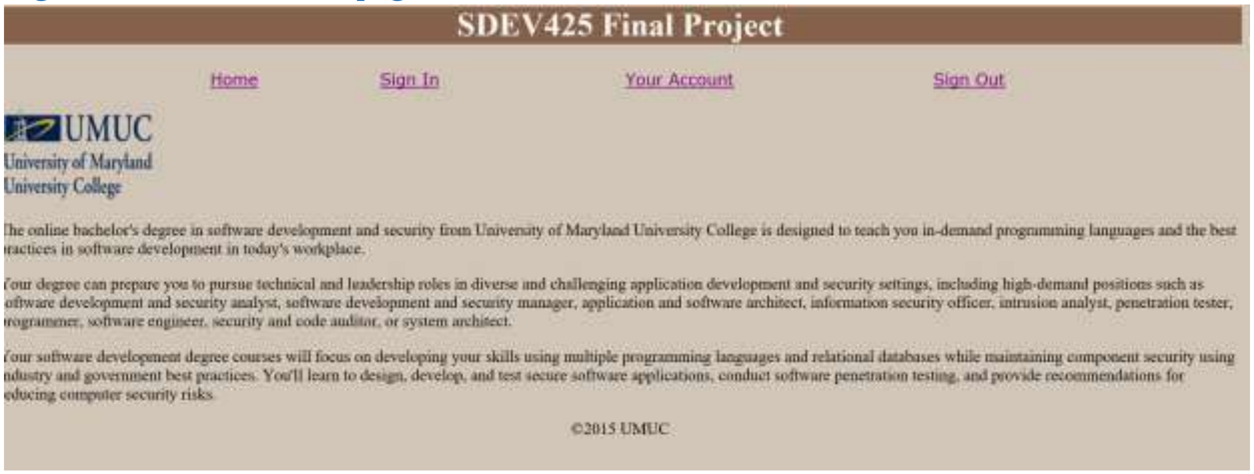


SDEV425 Final Project

Home Sign In Your Account Sign Out

Thank you for visiting our Web Site.

Figure 1.7 Back to home page



SDEV425 Final Project

Home Sign In Your Account Sign Out

UMUC
University of Maryland
University College

The online bachelor's degree in software development and security from University of Maryland University College is designed to teach you in-demand programming languages and the best practices in software development in today's workplace.

Your degree can prepare you to pursue technical and leadership roles in diverse and challenging application development and security settings, including high-demand positions such as software development and security analyst, software development and security manager, application and software architect, information security officer, intrusion analyst, penetration tester, programmer, software engineer, security and code auditor, or system architect.

Your software development degree courses will focus on developing your skills using multiple programming languages and relational databases while maintaining component security using industry and government best practices. You'll learn to design, develop, and test secure software applications, conduct software penetration testing, and provide recommendations for reducing computer security risks.

©2015 UMUC

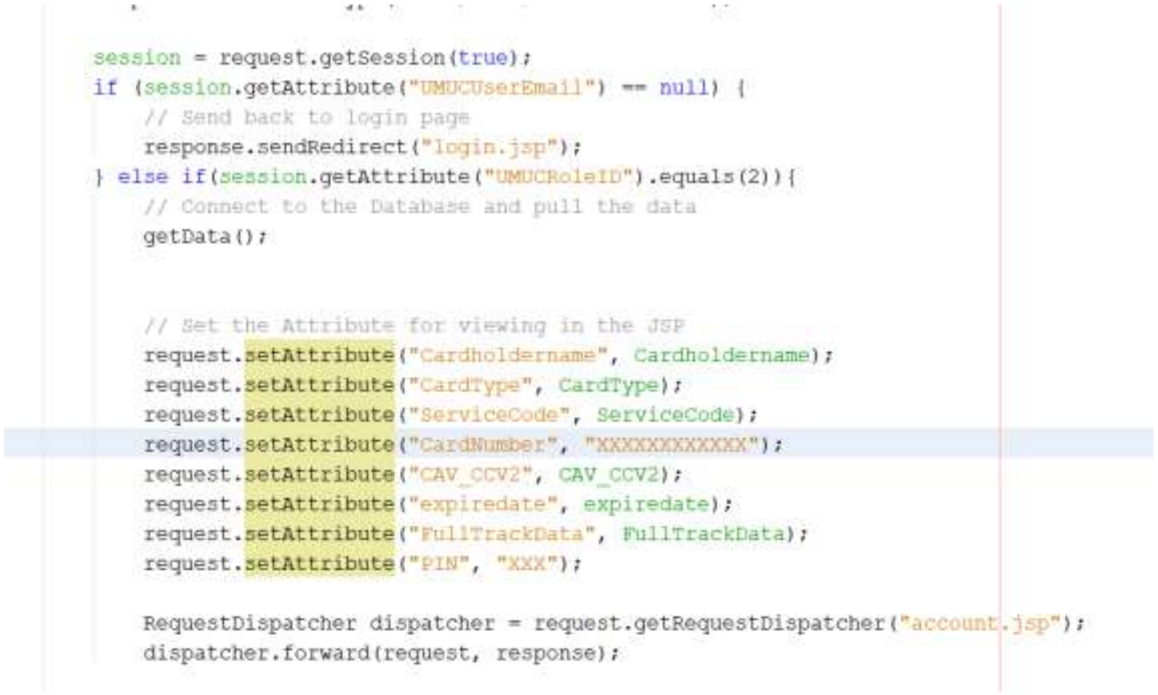
Part 2: PCI Compliance:

With the code application provide, I have made a few corrections to the code to meet PCI Compliance Requirements. Some requirements I felt did not require code correction but I provided information on how this requirement should be implemented and what developers would want to do beyond the application.

Figure 2.1 Requirement 3: Protect stored cardholder data

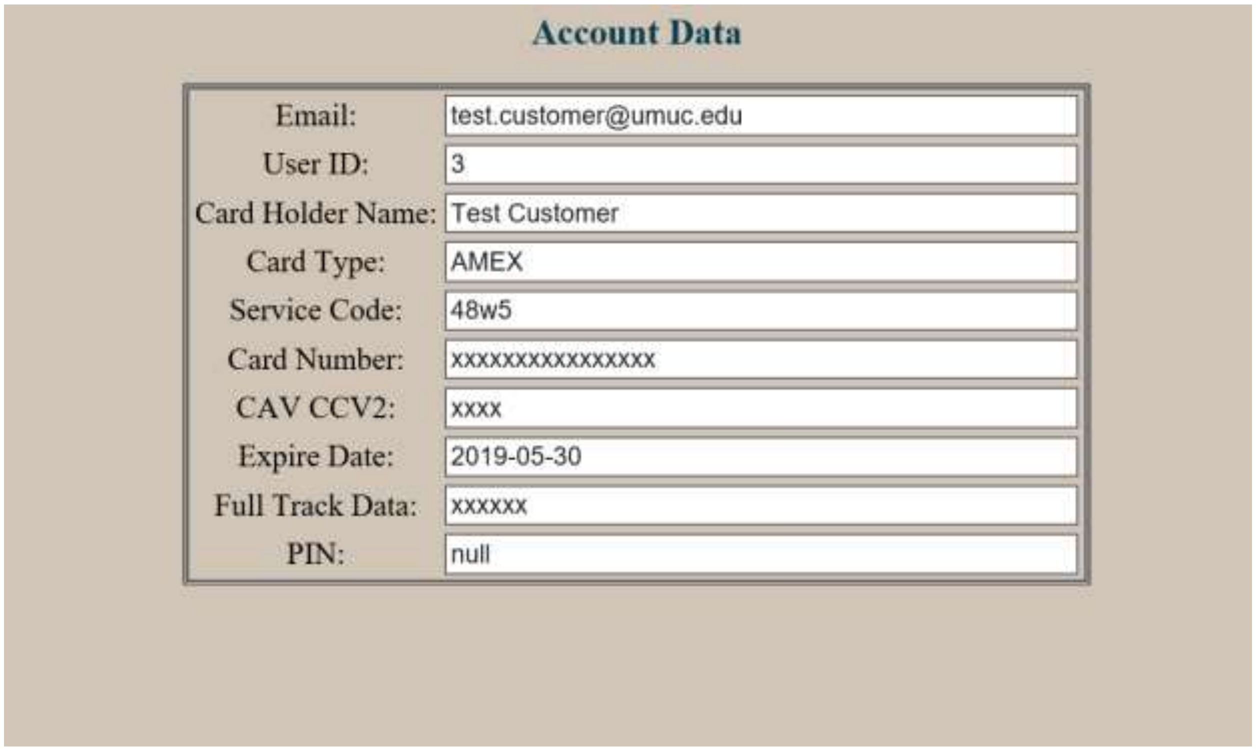
This requirement is set to create methods to protect stored card data. In 2020 software security is extremely important and as a developer one of a main jobs is to ensure the users sensitive information is protected at all times. Also this could lead to bigger issues for developers and companies if there application security is breeched. I decided to mask some of the users data based on role level. Blocking the customer's data helps prevent the user form accidently letting some of the user. For this requirement I focused on Requirement 3.2.3: Do not store personal Identification number and Requirement 3.3 Mask PAN when displayed (I masked the card number).

Figure 2.1.1: Requirement 3.3 Masking Script



```
session = request.getSession(true);
if (session.getAttribute("UMUCUserEmail") == null) {
    // Send back to login page
    response.sendRedirect("login.jsp");
} else if(session.getAttribute("UMUCRoleID").equals(2)){
    // Connect to the Database and pull the data
    getData();


    // Set the Attribute for viewing in the JSP
    request.setAttribute("Cardholdername", Cardholdername);
    request.setAttribute("CardType", CardType);
    request.setAttribute("ServiceCode", ServiceCode);
    request.setAttribute("CardNumber", "XXXXXXXXXXXX");
    request.setAttribute("CAV_CCV2", CAV_CCV2);
    request.setAttribute("expiredate", expiredate);
    request.setAttribute("FullTrackData", FullTrackData);
    request.setAttribute("PIN", "XXX");

    RequestDispatcher dispatcher = request.getRequestDispatcher("account.jsp");
    dispatcher.forward(request, response);
```

Figure 2.1.1: Requirement 3.3 Masking Results



Account Data

Email:	test.customer@umuc.edu
User ID:	3
Card Holder Name:	Test Customer
Card Type:	AMEX
Service Code:	48w5
Card Number:	xxxxxxxxxxxxxxxx
CAV CCV2:	xxxx
Expire Date:	2019-05-30
Full Track Data:	xxxxxx
PIN:	null

Figure 2.1.2: Requirement 3.3

Figure 2.2 Requirement 4: Encrypt Transmission of Cardholder Data across open public network

Sensitive information must be protected at all times while the user is using your web application. One way to ensure is to ensure you are hosting the application over an HTTPS host. This way the application is using a strong cryptography and security protocol such as TLS or SSH.

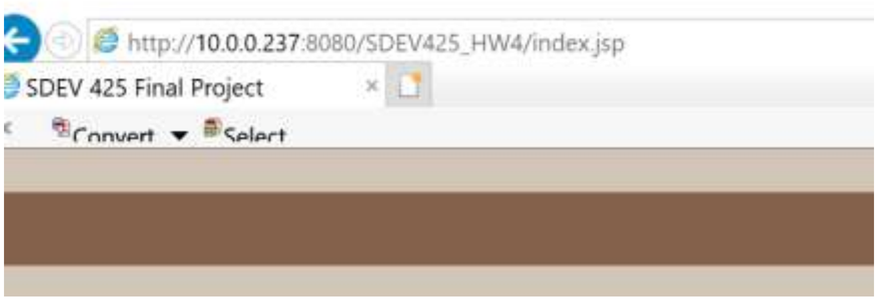


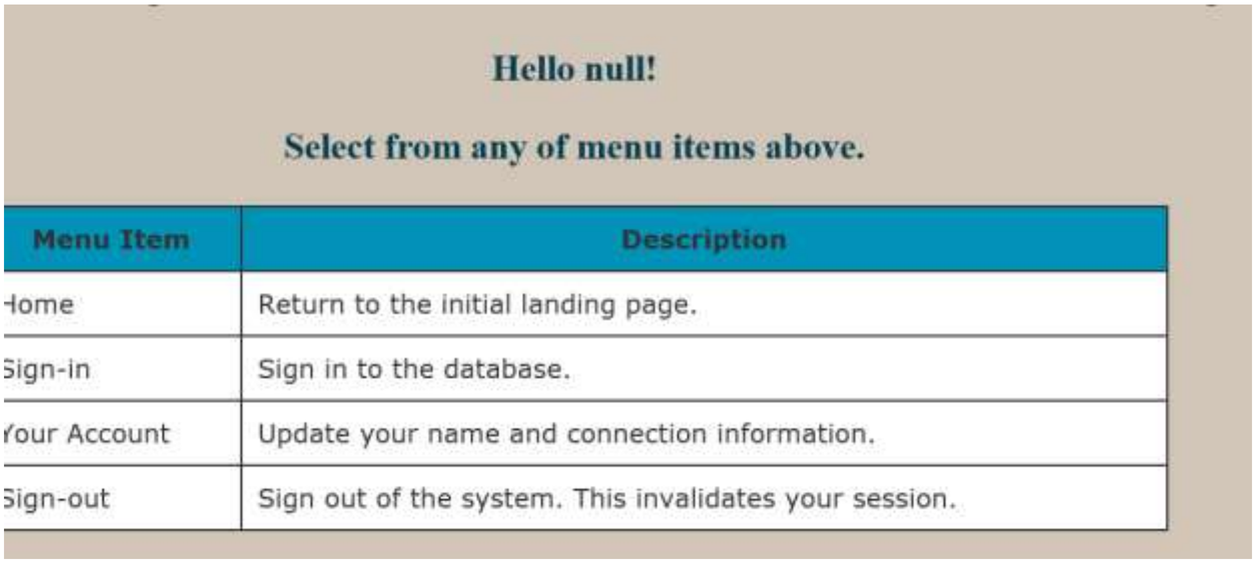
Figure 2.3 Requirement 5:

The purpose of this requirement is to ensure proper anti-virus is being used on the user computer and the developer's computer. This is not something we are able to implement into the application.

Figure 2.4 Requirement 6:

For this requirement I focused on 6.5.8 Improper Access control, one of the issues is being able to access the welcome.jsp site without actually logging in. I fixed this by redirecting the user back to the index.jsp page if email = null.

Fig 2.4.2: Screen shot of error

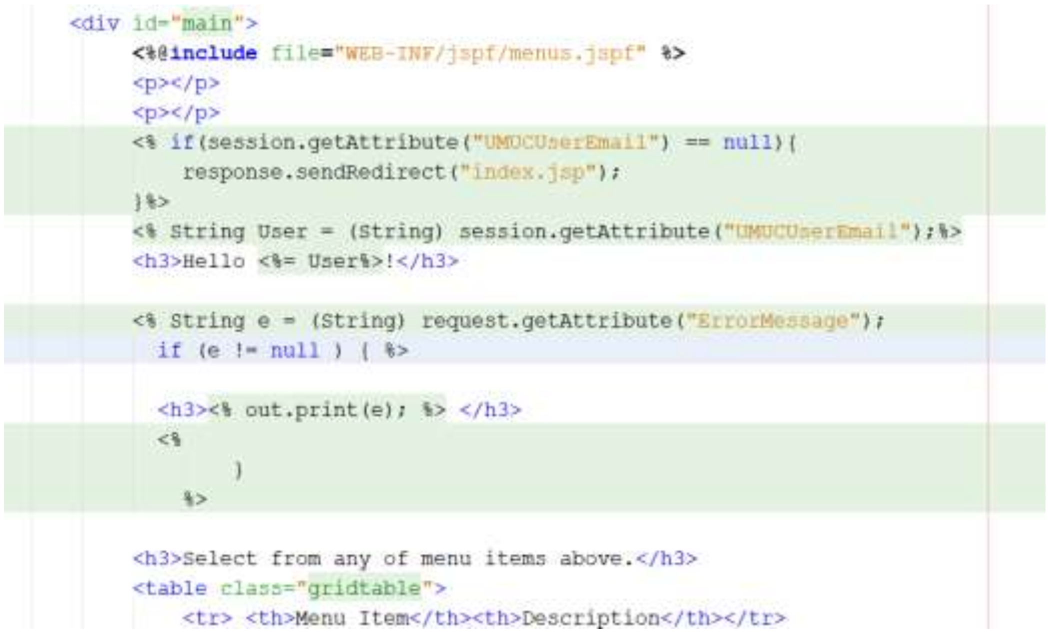


Hello null!

Select from any of menu items above.

Menu Item	Description
Home	Return to the initial landing page.
Sign-in	Sign in to the database.
Your Account	Update your name and connection information.
Sign-out	Sign out of the system. This invalidates your session.

Fig 2.4.1: Script correction

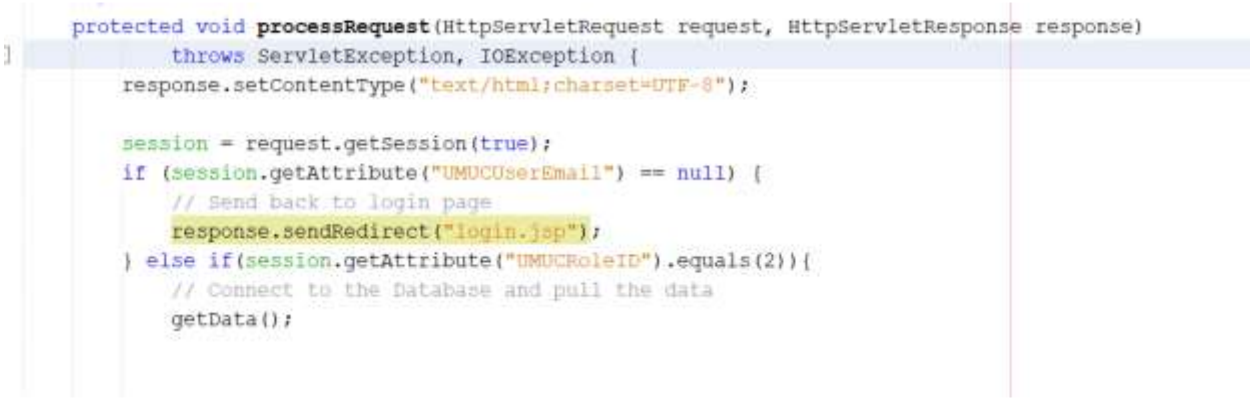


```
<div id="main">
    <%@include file="WEB-INF/jspf/menus.jspf" %>
    <p></p>
    <p></p>
    <% if(session.getAttribute("UMUCUserEmail") == null){
        response.sendRedirect("index.jsp");
    }%>
    <% String User = (String) session.getAttribute("UMUCUserEmail");%>
    <h3>Hello <%= User%>!</h3>

    <% String e = (String) request.getAttribute("ErrorMessage");
      if (e != null ) { %>

      <h3><% out.print(e); %> </h3>
      <%
            }
        %>

    <h3>Select from any of menu items above.</h3>
    <table class="gridtable">
        <tr> <th>Menu Item</th><th>Description</th></tr>
```

Figure 2.5 Requirement 7: Restrict access to cardholder data by business need to know:

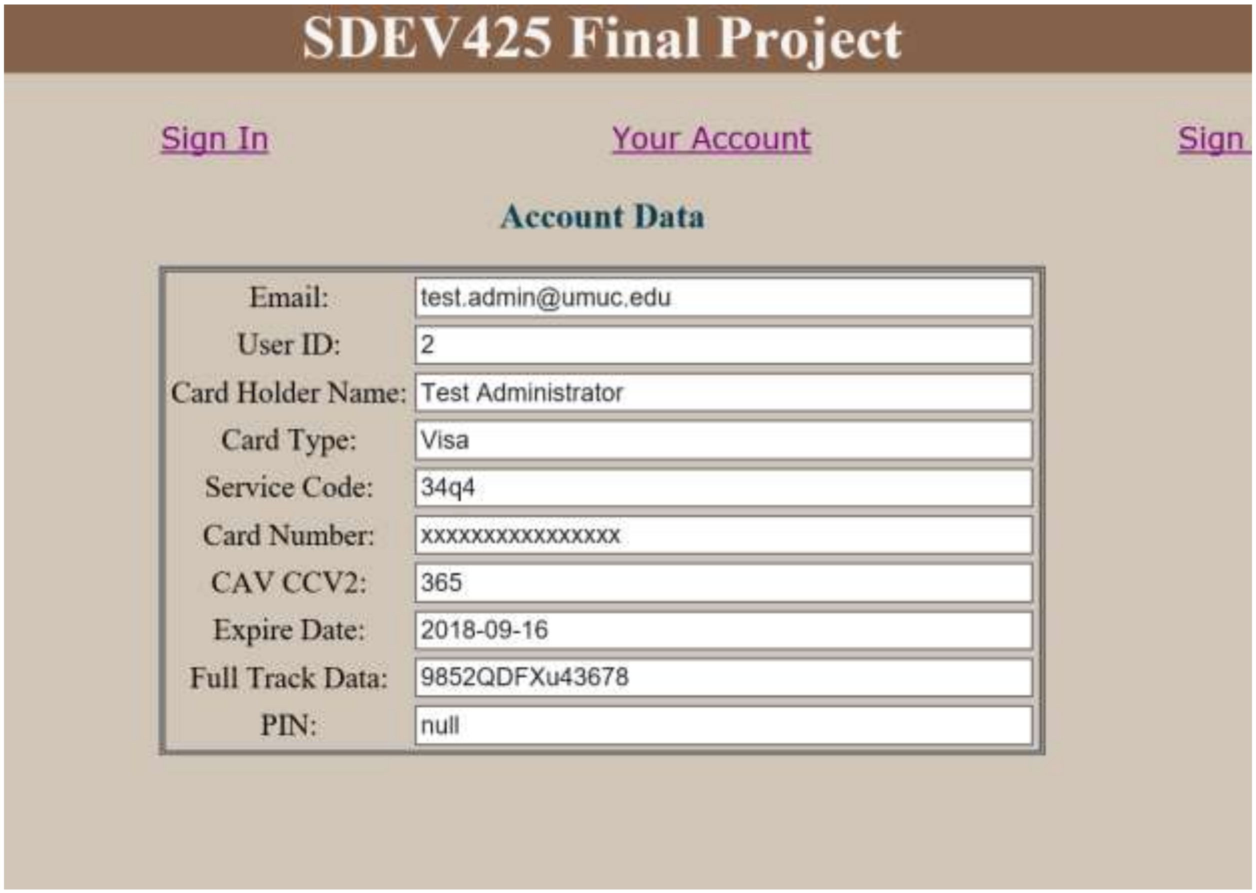
Full Track Data is not needed to be known by the customer, this should be information that should be only accessed by admin. So what I decided to do is only provide the customer with select information based on their role ID. Sense all user will be assigned a role id of 1 or 2. This will dictate if they are an admin or a regular user. Those who are admin would be able to see all information. Regular user will see information we allow them to see. Based on what I implemented I focused on Compliance 7.1.1:

Fig: 2.5.1: Script



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");

    session = request.getSession(true);
    if (session.getAttribute("UMUCUserEmail") == null) {
        // Send back to login page
        response.sendRedirect("login.jsp");
    } else if(session.getAttribute("UMUCRoleID").equals(2)){
        // Connect to the Database and pull the data
        getData();
```

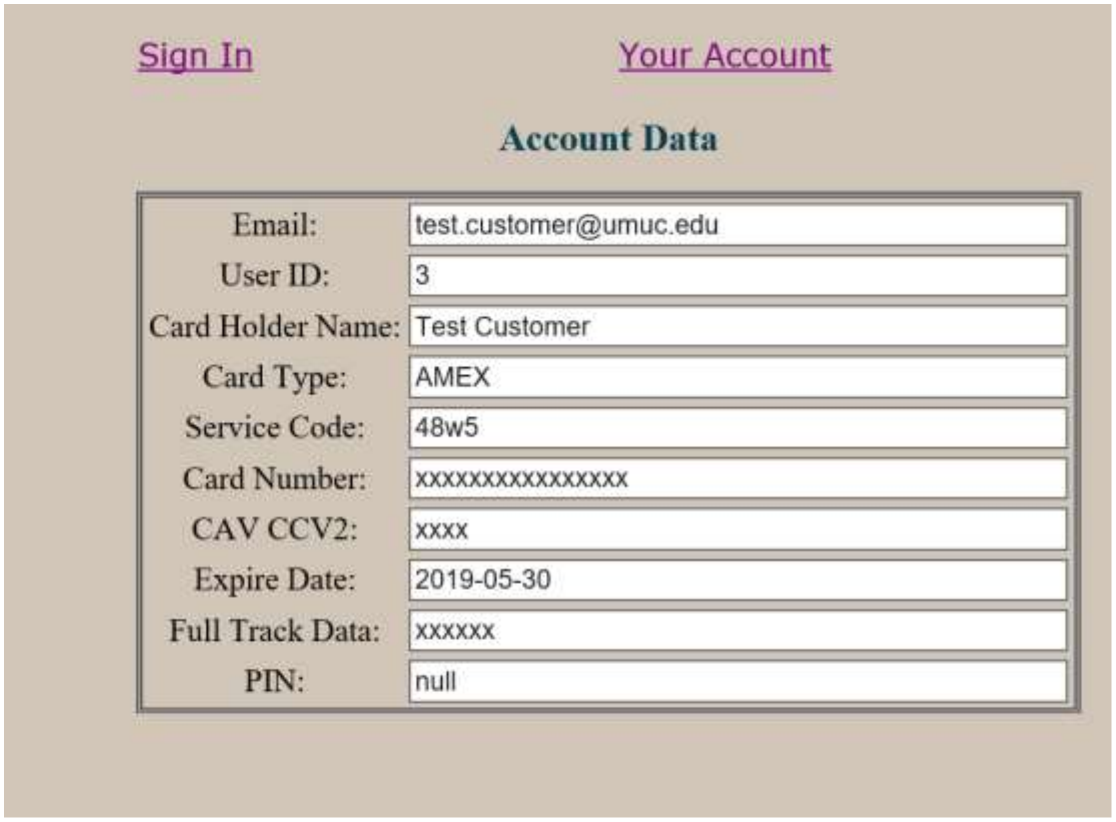
Fig: 2.5.2: Result



SDEV425 Final Project

Account Data

Email:	test.admin@umuc.edu
User ID:	2
Card Holder Name:	Test Administrator
Card Type:	Visa
Service Code:	34q4
Card Number:	xxxxxxxxxxxxxxxx
CAV CCV2:	365
Expire Date:	2018-09-16
Full Track Data:	9852QDFXu43678
PIN:	null



Sign In Your Account

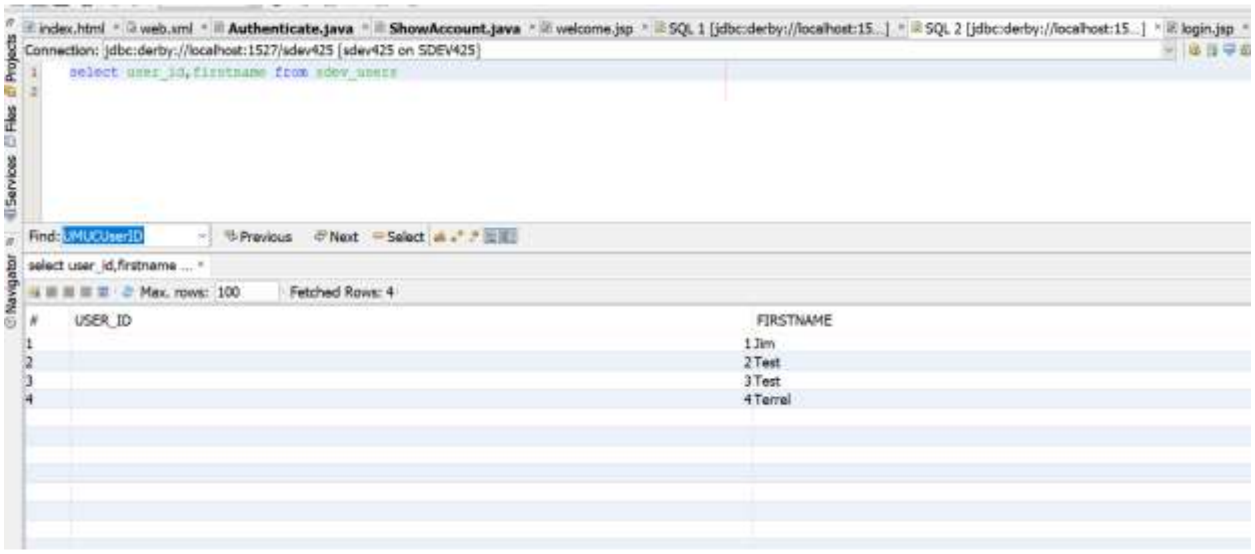
Account Data

Email:	test.customer@umuc.edu
User ID:	3
Card Holder Name:	Test Customer
Card Type:	AMEX
Service Code:	48w5
Card Number:	xxxxxxxxxxxxxxxx
CAV CCV2:	xxxx
Expire Date:	2019-05-30
Full Track Data:	xxxxxx
PIN:	null

Figure 2.6 Requirement 8:

For Requirement 8 I focused on Compliance 8.1.1, this was already implemented when we uploaded the data to our SQL server. Each users has a unique account ID. With having this unique ID, we are able to determine who accessed what and when.

Fig 2.6.1 Results:



Connection: jdbc:derby://localhost:1527/sdev425 [sdev425 on SDEV425]

```
select user_id,firstname from sdev_users
```

Find: UMUCUserID

select user_id,firstname ...

Max. rows: 100 Fetched Rows: 4

#	USER_ID	FIRSTNAME
1	1	Jim
2	2	Test
3	3	Test
4	4	Terrel

Figure 2.7 Requirement 9:

This requirement focuses on physical access to users card informaiton, for this application we would need to simply lock my computer once I walk away from my computer. This is also a standard for most organzations that deal with sensetive information on a daily basis. For example at my previous job I worked with VA benefits and this included Veterans PII.

Refernece:

Tutorialspoint.com. (n.d.). SQL Tutorial. Retrieved July 29, 2019, from https://www.tutorialspoint.com/sql/

Working with the Java DB (Derby) Database. (n.d.). Retrieved July 28, 2019, from https://netbeans.org/kb/docs/ide/java-db.html