INFO 6250 – Web Development Tools and Methods

Fall 2021, Northeastern University

Project Report

W.A.M.P.

The Web Accessible Media Player Application

Project By

Name: Asfan Sajid | NUID:001546755

Title: **W.A.M.P.** – The Web Accessible Media Player Application

1. Summary:

A web accessible media player based on the playlist model, where users can search for media and play it, search for a user, and play their playlist and what's more, they can add it to their own personal playlist too. WAMP uses Web's Spring MVC Architecture incorporating and leveraging the Hibernate and DAO frameworks. It allows the media managers(admins) to add media to the application to which the users can groove to. Additionally, it allows the users and the admins to update their respective personal details. Whatmore? Do not like the application? You can delete your trace from it with a simple click.

2. Roles and Functionalities:

- **Administrator (Admin)**:
  - Register as an Admin (Need to be an administrator to register as admin for the application).
  - Additionally, can use the same credentials to log in as a user too.
  - Login with their credentials and update their profile information.
  - Can manage the media in the database by adding or deleting media.
  - Additionally, can add new artists to the base and can associate an artist with the media uploaded.
  - Can delete artist(s) from the application, consequentially deleting the media associated with the artist as well.
  - Can only delete the user(s) from the application.
  - Can delete their account from the application.

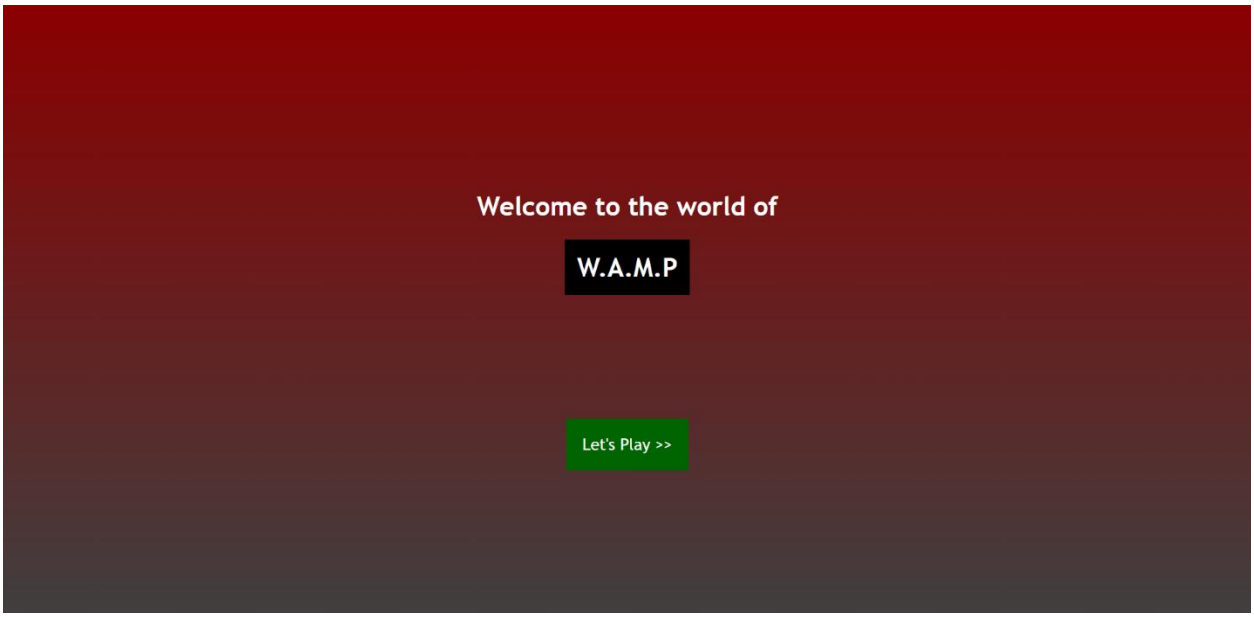
- **User (User)**:
  - Register as a User for the application.
  - Login with their credentials and update their profile information.
  - Can search for other users and view their respective playlists.
  - Can add media from another user's playlist to their own playlist.
  - Can search for media from the media search and play it before adding it to the playlist.
  - Can delete media from their playlist too.
  - Can update their personal information in the application.
  - Can delete their account from the application.

- **Artist (Virtual Role)**:
  - When deleted, deleted the media associated with the respective artist.

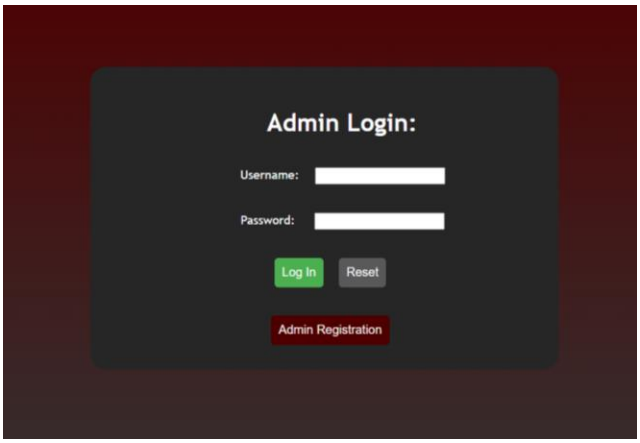
Functionalities:

1. Login Page – Allows the user to login into the WAMP application with their credentials
2. Search Page – Allows the user to search for other users and media to stream
3. Account Page – Allows the user to update the account details and delete the account.
4. User Signup Page - Allows the user to sign up for the WAMP application with their credentials
5. User Dashboard Page - Presents the user with the search media, logout functionalities of the WAMP application.
6. Admin Login Page – Allows the admin to login into the WAMP application with their credentials
7. Admin Management Page – Allows the Admin to search for other users, media, and artists to delete or manage their respective lists.
8. Admin Account Page – Allows the admin to update the account details and delete the account.
9. Admin Signup Page - Allows the admin to sign up for the WAMP application with their credentials
10. Admin Dashboard Page - Presents the admin with the management of media, users, artists, and logout functionalities of the WAMP application.

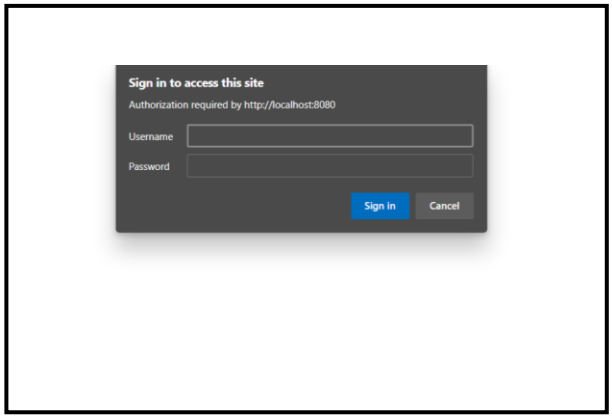
3. **Screenshots**:

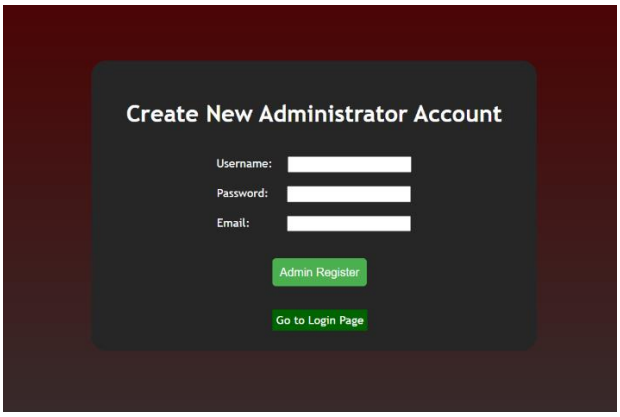


1. Home Page

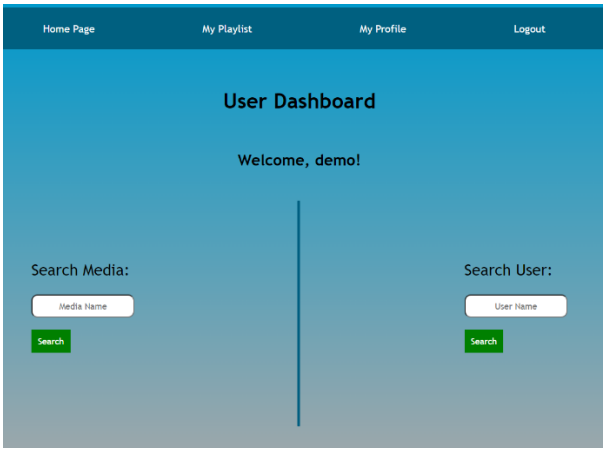


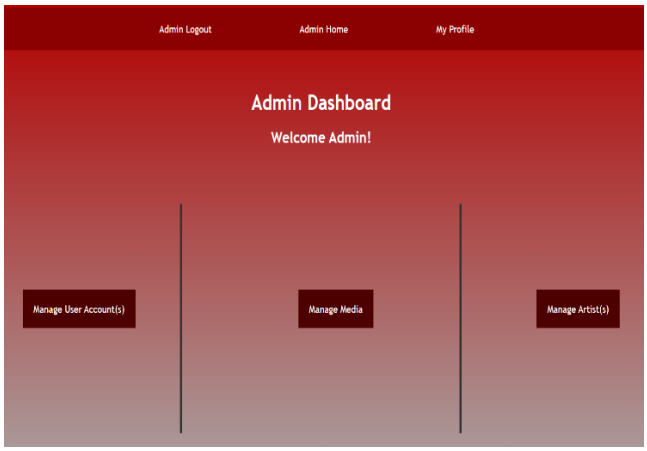
2. Login Page(s)



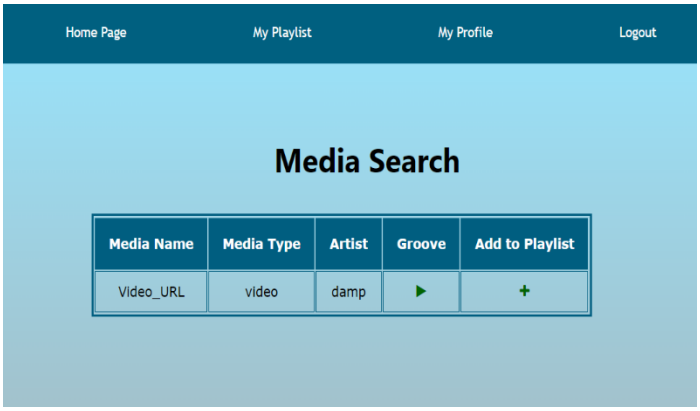


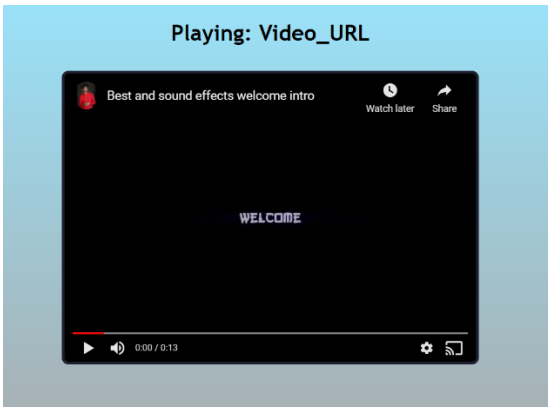
3. Declarative Security (Admin Registration)
(*Idea: Only an application admin can register as admin*)

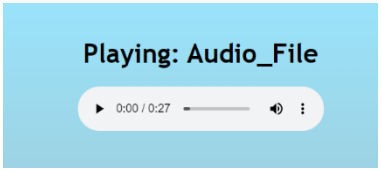




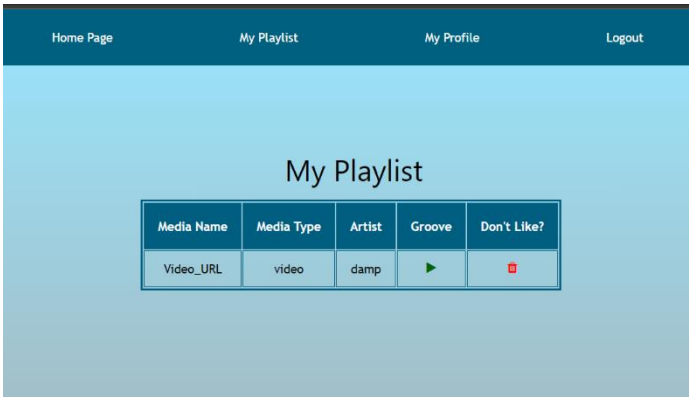
4.Dashboards



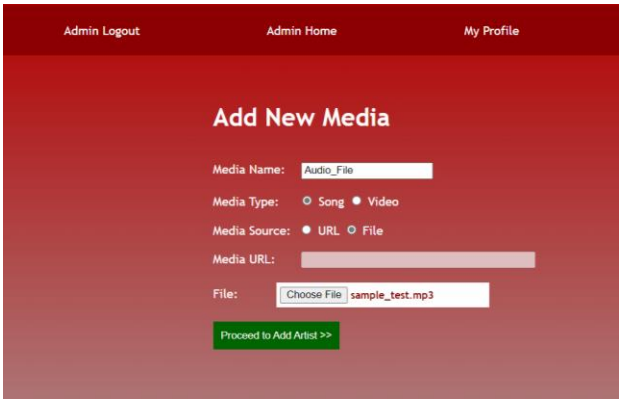


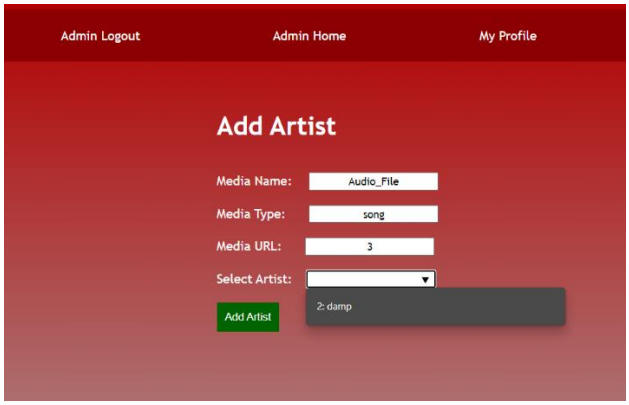


5.Search Media and Embedded Media Player

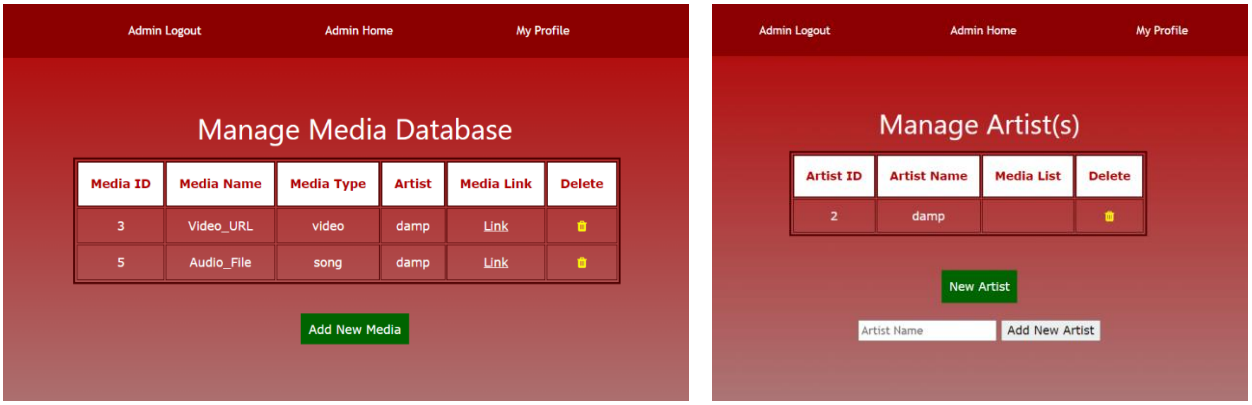


6. User's Playlist

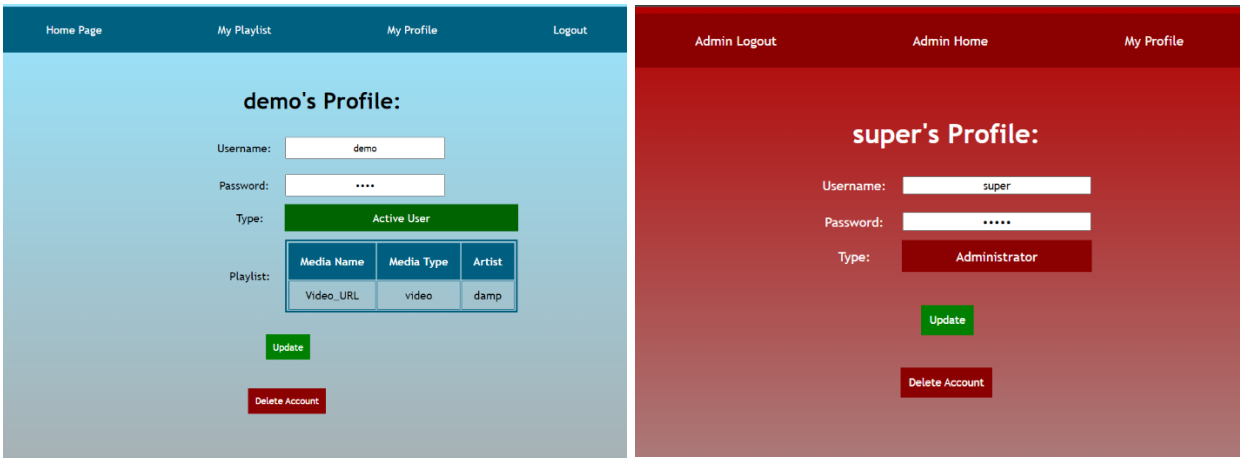




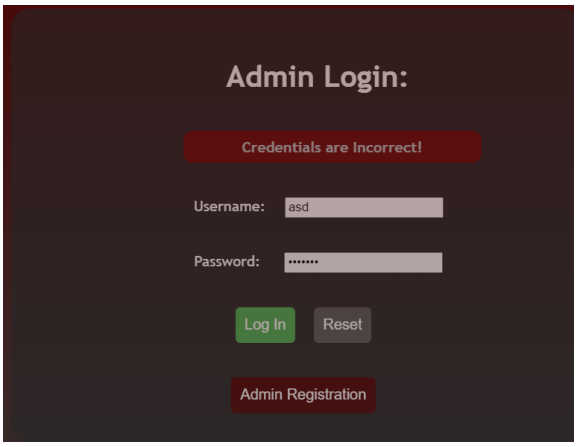
7. Admin – Add Media and Artist

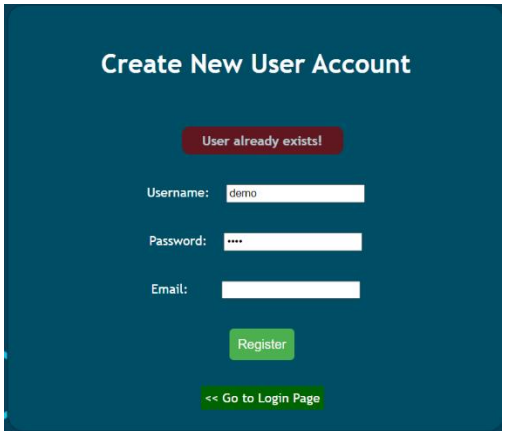


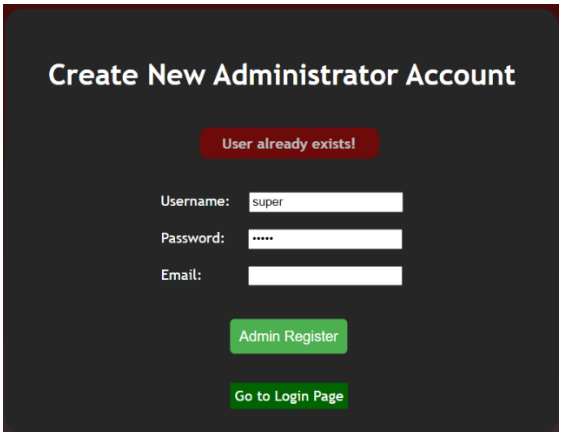
8. Admin – Manage Media and Artist

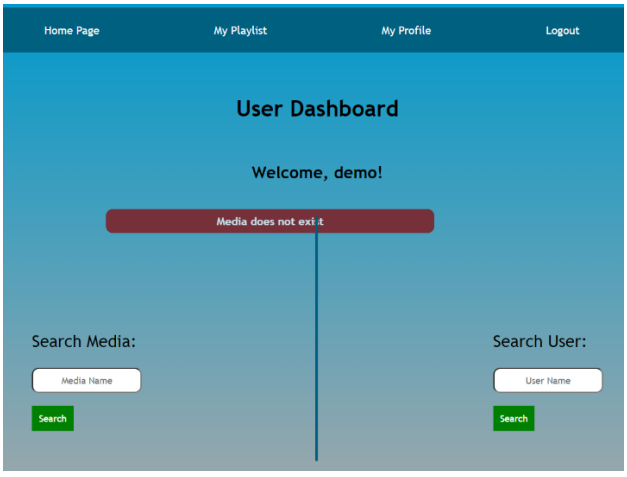


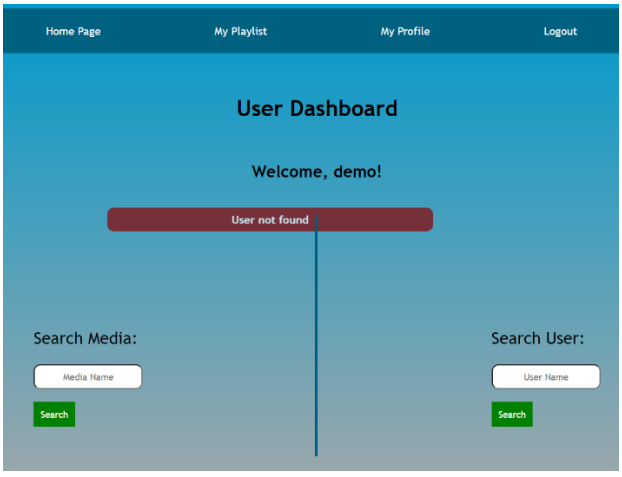
9. User, Admin Update Profiles



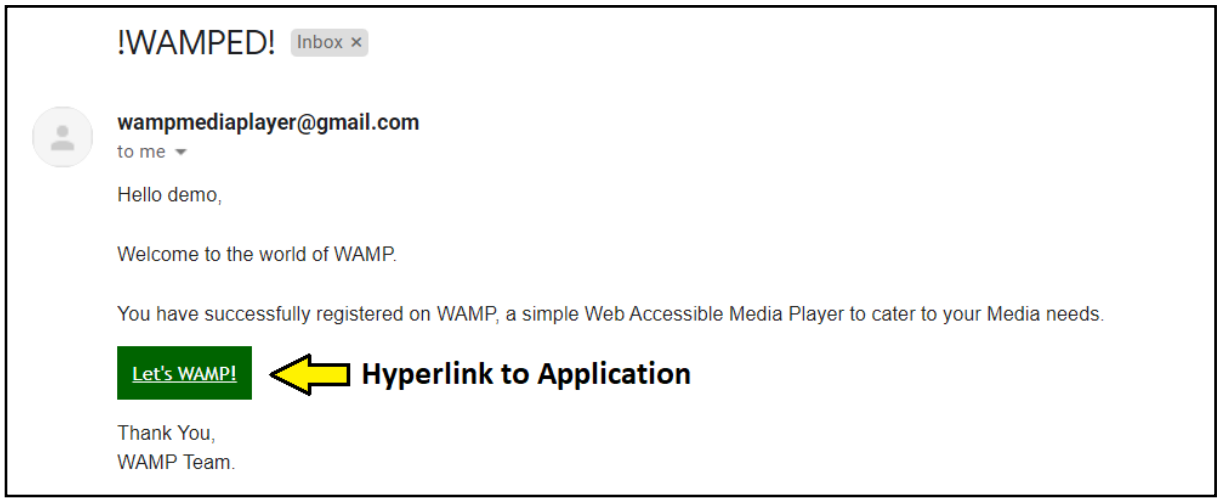




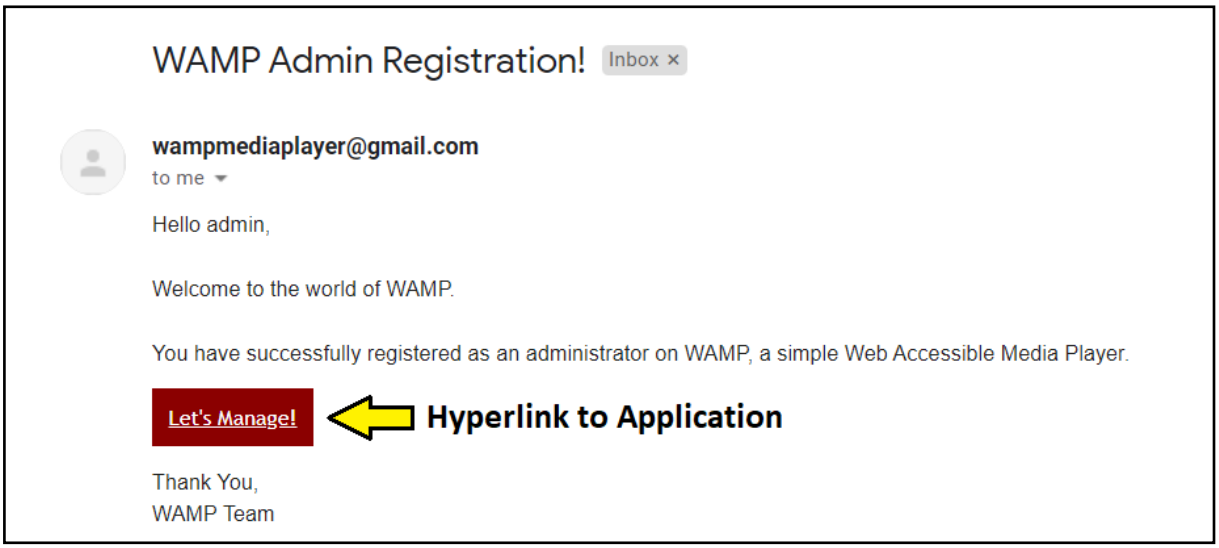




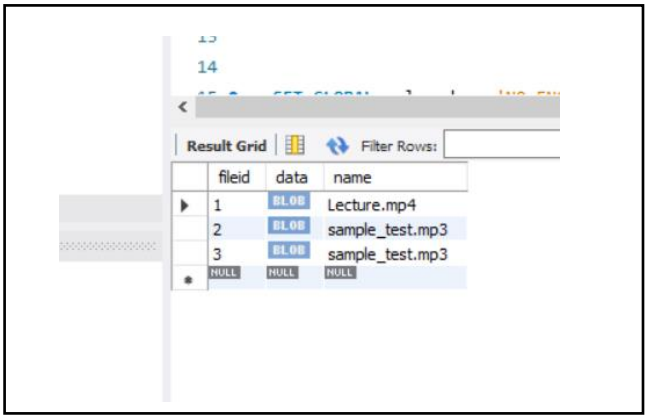
10. Validations



11. User Registration Email



12. Admin Registration Email



13. Database Media Storage

4. Technologies Used

- Spring Model View Controller (MVC) Architecture
- Hibernate ORM - Object Relational Mapping
- DAO – Data Access Object Pattern
- Apache Commons – Multipart File Handler
- Javax Mailer: Email API
- iFrame API – Media Player Embedding
- JSP – Java Server Pages
- JavaScript – Client-Side Logic Handling
- HTML – Hyper Text Markup Language
- CSS – Cascading Style Sheets
- STS – Spring Tool Suite
- Java – v15
- Apache Tomcat – v9
- MySQL – MySQL Database v8
- Apache NetBeans

5. Appendix:

5.1 Controllers – Source Code:

1. Admin Controller:

```
package com.me.wamp.controller;

import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.support.SessionStatus;
import org.springframework.web.multipart.commons.CommonsMultipartFile;

import com.me.wamp.dao.ArtistDAO;
import com.me.wamp.dao.FileDAO;
import com.me.wamp.dao.MediaDAO;
import com.me.wamp.dao.PlaylistDAO;
import com.me.wamp.dao.UserDAO;
import com.me.wamp.pojo.Artist;
import com.me.wamp.pojo.File;
import com.me.wamp.pojo.Media;
import com.me.wamp.pojo.Playlist;
import com.me.wamp.pojo.User;
import com.me.wamp.validation.AdminValidator;
import com.me.wamp.validation.UserValidator;

import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;

@Controller
public class AdminController {

	@Autowired
	AdminValidator adminValidator;

	public AdminController() {

	}

	// Display Admin error form
	@GetMapping("/error.htm")
	public String showError(HttpServletRequest request) {
		return "admin-error";
	}

	// display Admin login form
	@GetMapping("/admin/login.htm")
	public String showLogin(ModelMap model) {
		User admin = new User();
		model.addAttribute("admin", admin);
		return "admin-login";
```

```
	}

	@PostMapping("/admin/login.htm")
	public String handleLoginForm(@ModelAttribute("admin") User admin,
BindingResult result, SessionStatus status,
			HttpServletRequest request, ModelMap map, UserDAO userdao) {
		if (result.hasErrors()) {
			return "user-login";
		} else {
			HttpSession session = request.getSession(true);
			User a = userdao.getUser(admin);
			if (a != null) {
				session.setAttribute("currentAdmin", a);
				status.setComplete();
				return "admin-dashboard";
			} else {
				request.setAttribute("error", "Incorrect Admin Credentials");
				return "admin-login";
			}
		}
	}

	@GetMapping("/admin/save.htm")
	public String saveAdminGet(ModelMap model) {
		User admin = new User(); // FormBackingObject
		model.addAttribute("admin", admin);
		return "admin-registration";
	}

	@PostMapping("/admin/save.htm")
	public String saveAdminPost(@ModelAttribute("admin") User admin,
BindingResult result, SessionStatus status,
			UserDAO userdao, PlaylistDAO playlistdao, HttpServletRequest
request) {
		String view = null;
		if (result.hasErrors())
			view = "admin-login";
		else {
			User a = userdao.getUser(admin);
			if (a == null) {
				admin.setRole("admin");

				Playlist playlist = new Playlist();
				playlist.setName(admin.getUsername() + "_playlist");
				playlist.setType("song");
				playlistdao.createPlaylist(playlist);
```

```
                                admin.setPlaylist(playlist);

                                userdao.createUser(admin);

                                sendEmail(request.getParameter("email"),
admin.getUsername());

                                status.isComplete();
                                view = "admin-registrationSuccess";
                        } else {
                                request.setAttribute("error", "Admin already exists");
                                view = "admin-registration";
                        }
                }
                return view;
        }

        public void sendEmail(String email, String user) {
                String to = email;
                final String from = "wampmediaplayer@gmail.com";
                final String password = "Password@123";

                Properties props = new Properties();
                props.put("mail.smtp.host", "smtp.gmail.com");
                props.put("mail.smtp.socketFactory.port", "465");
                props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
                props.put("mail.smtp.auth", "true");
                props.put("mail.smtp.port", "465");
                Session session = Session.getDefaultInstance(props, new
javax.mail.Authenticator() {
                        protected PasswordAuthentication getPasswordAuthentication() {
                                return new PasswordAuthentication(from, password);
                        }
                });

                try {
                        MimeMessage message = new MimeMessage(session);
                        message.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));
                        message.setSubject("WAMP Admin Registration!");
                        message.setText("Hello " + user
                                        + ", <br><br>Welcome to the world of
WAMP.<br><br>You have successfully registered as an administrator on WAMP, a
simple Web Accessible Media Player.<br><br> <a style=\"background-color: darkred;
padding: 10px; color: white; font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida
Grande', 'Lucida Sans', Arial, sans-serif;\"
```

```
href=\"http://localhost:8080/wamp/admin/login.htm\" target=\"_blank\">Let's
Manage!</a><br><br>Thank You,<br>WAMP Team",
                                        "UTF-8", "html");

                        // Send message
                        Transport.send(message);
                } catch (Exception mex) {
                        mex.printStackTrace();
                }
        }

        // display admin dashboard
        @GetMapping("/admin/dashboard.htm")
        public String showDashboard(HttpServletRequest request) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentAdmin")!=null) {
                        return "admin-dashboard";
                } else
                        return "home";
        }

        // display all users accounts
        @GetMapping("/admin/user.htm")
        public String manageUserAccount(HttpServletRequest request, UserDAO
userdao) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentAdmin")!=null) {

                        List<User> userList = userdao.getAllUsers();

                        session.setAttribute("userList", userList);

                        return "admin-userManagement";
                } else
                        return "home";
        }

        @GetMapping("/admin/artist.htm")
        public String manageArtist(HttpServletRequest request, ArtistDAO artistdao) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentAdmin")!=null) {

                        List<Artist> artistList = artistdao.getAllArtists();

                        session.setAttribute("artistList", artistList);

                        return "admin-artistManagement";
```

```
		} else
			return "home";
	}

	@GetMapping("/admin/artist/add.htm")
	public String artistAdd(HttpServletRequest request, ArtistDAO artistdao) {
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentAdmin")!=null) {

			Artist artist = new Artist();

			artist.setName(request.getParameter("artistName"));

			artistdao.createArtist(artist);

			return "admin-dashboard";
		} else
			return "home";
	}

	@GetMapping("/admin/artist/delete.htm")
	public String deleteArtistAdmin(MediaDAO mediadao, ArtistDAO artistdao,
PlaylistDAO playlistdao,
			HttpServletRequest request) {
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentAdmin")!=null) {

			int artistid = Integer.parseInt(request.getParameter("artistid"));
			Artist artist = artistdao.getArtist(artistid);

			List<Media> mediaList = mediadao.getAllMedia();

			for (Media m : mediaList) {
				if (m.getArtist().equals(artist)) {
					for (Playlist p : playlistdao.getAllPlaylists()) {
						p.removeMedia(m);
						playlistdao.updatePlaylist(p);
					}
					mediadao.deleteMedia(m);
				}
			}

			artistdao.deleteArtist(artist);

			session.setAttribute("artistList", artistdao.getAllArtists());
			session.setAttribute("mediaList", mediadao.getAllMedia());
```

```
                return "admin-dashboard";
            } else
                return "home";
        }

        @GetMapping("/userAdmin/delete.htm")
        public String deleteUserAdmin(UserDAO userdao, PlaylistDAO playlistdao,
HttpServletRequest request) {
            HttpSession session = request.getSession(true);
            if (session.getAttribute("currentAdmin")!=null) {

                String userid = request.getParameter("userid");
                User user = userdao.getUserById(Integer.parseInt(userid));

        playlistdao.deletePlaylist(playlistdao.getPlaylist(user.getPlaylist().getPlaylistid()));
                userdao.deleteUser(user);
                return "admin-user-delete";
            } else
                return "home";
        }

        // display all media
        @GetMapping("/admin/media.htm")
        public String manageMedia(HttpServletRequest request, MediaDAO mediadao) {
            HttpSession session = request.getSession(true);
            if (session.getAttribute("currentAdmin")!=null) {

                List<Media> mediaList = mediadao.getAllMedia();

                session.setAttribute("mediaList", mediaList);

                return "admin-mediaManagement";
            } else
                return "home";
        }

        @GetMapping("/admin/media/delete.htm")
        public String deleteMediaAdmin(MediaDAO mediadao, ArtistDAO artistdao,
HttpServletRequest request) {
            HttpSession session = request.getSession(true);
            if (session.getAttribute("currentAdmin")!=null) {

                String mediaid = request.getParameter("mediaid");
                Media media =
mediadao.getMediaById(Integer.parseInt(mediaid));
                mediadao.deleteMedia(media);
```

```
                return "admin-media-delete";
        } else
                return "home";
    }

    @GetMapping("/admin/addMedia/add.htm")
    public String addMedia(HttpServletRequest request) {
        HttpSession session = request.getSession(true);
        if (session.getAttribute("currentAdmin")!=null) {

                return "admin-media-add";
        } else
                return "home";
    }

    @PostMapping("/admin/media/add.htm")
    public String addMedia(HttpServletRequest request, MediaDAO mediadao,
ArtistDAO artistdao) throws Exception {
        HttpSession session = request.getSession(true);
        if (session.getAttribute("currentAdmin")!=null) {

                Media media = new Media();
                System.out.println("URL");
                String mediaName = request.getParameter("mediaName");
                media.setName(mediaName);

                String mediaType = request.getParameter("mediaType");
                media.setType(mediaType);

                System.out.println(request.getParameter("mediaUrl"));
                String mediaUrl = request.getParameter("mediaUrl");
                media.setUrl(mediaUrl);

                List<Artist> artistList = artistdao.getAllArtists();

                session.setAttribute("media", media);
                session.setAttribute("artistList", artistList);

                return "admin-media-addArtist";
        } else
                return "home";
    }

    @PostMapping("/admin/media/addFile.htm")
    public String addMedia(HttpServletRequest request, MediaDAO mediadao,
ArtistDAO artistdao, FileDAO filedao,
```

```
                        @RequestParam("fileUpload") CommonsMultipartFile[] fileUpload)
throws Exception {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentAdmin")!=null) {

                        Media media = new Media();
                        System.out.println("File");
                        String mediaName = request.getParameter("mediaName");
                        media.setName(mediaName);

                        String mediaType = request.getParameter("mediaType");
                        media.setType(mediaType);

                        if (request.getParameter("mediaUrl") != null) {
                                media.setUrl(request.getParameter("mediaUrl"));
                        } else {
                                int fileid = 0;
                                try {

        System.out.println(request.getParameter("fileUpload"));
                                        fileid = handleFileUpload(fileUpload, filedao);
                                        media.setUrl(Integer.toString(fileid));
                                } catch (Exception e) {
                                        System.out.println("Failed to upload Media");
                                        System.out.println(e.getStackTrace());
                                }
                        }

                        List<Artist> artistList = artistdao.getAllArtists();

                        session.setAttribute("media", media);
                        session.setAttribute("artistList", artistList);

                        return "admin-media-addArtist";
                } else
                        return "home";
        }

        public int handleFileUpload(CommonsMultipartFile[] fileUpload, FileDAO filedao)
throws Exception {
                if (fileUpload != null && fileUpload.length > 0) {
                        for (CommonsMultipartFile aFile : fileUpload) {
                                System.out.println("Saving file: " +
aFile.getOriginalFilename());
                                File uploadFile = new File();
                                uploadFile.setName(aFile.getOriginalFilename());
                                uploadFile.setData(aFile.getBytes());
```

```
                    return (filedao.createFile(uploadFile));
                }
            }
            return 0;
        }

        @GetMapping("/admin/mediaAdd.htm")
        public String addMediaToDb(HttpServletRequest request, MediaDAO mediadao,
ArtistDAO artistdao,
                    PlaylistDAO playlistdao) {

            HttpSession session = request.getSession(true);
            if (session.getAttribute("currentAdmin")!=null) {

                Media media = (Media) session.getAttribute("media");

                int mediaid = 0;
                String artistid = (String) request.getParameter("artist");
                Artist artist = null;

                if (artistid.split(":").length > 1) {
                    artistid = artistid.split(":")[0];
                    artist = artistdao.getArtist(Integer.parseInt(artistid));
                }

                if (artist == null) {
                    artist = new Artist();
                    artist.setName(request.getParameter("artist"));


        media.setArtist(artistdao.getArtist(artistdao.createArtist(artist)));

                    mediaid = mediadao.createMedia(media);
                } else {
                    media.setArtist(artist);

                    mediaid = mediadao.createMedia(media);
                }
                Media updateMedia = mediadao.getMediaById(mediaid);
                updateMedia.setArtist(artist);
                mediadao.updateMedia(updateMedia);

                return "admin-dashboard";
            } else
                return "home";
        }
```

```
	@GetMapping("/admin/delete.htm")
	public String deleteUser(UserDAO userdao, PlaylistDAO playlistdao,
HttpServletRequest request, ModelMap model) {
		HttpSession session = request.getSession(true);
		if (session != null) {
			User admin = (User) session.getAttribute("currentAdmin");
			admin = userdao.getUserById(admin.getUserid());

	playlistdao.deletePlaylist(playlistdao.getPlaylist(admin.getPlaylist().getPlaylistid())
);
			userdao.deleteUser(admin);
			session.invalidate();
		}
		User user = new User();
		model.addAttribute("user", user);
		return "user-login";
	}

	@GetMapping("/admin/update.htm")
	public String updateAdminGet(HttpServletRequest request) {
		// Update the admin instance using the DAO class
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentAdmin")!=null) {

			return "admin-update";
		} else
			return "home";
	}

	@PostMapping("/admin/update.htm")
	public String updateUserPost(@ModelAttribute("user") User admin,
BindingResult result, SessionStatus status,
			UserDAO userdao, HttpServletRequest request) {
		// Update the user instance using the DAO class
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentAdmin")!=null) {

			User a = (User) session.getAttribute("currentAdmin");

			a.setUsername(request.getParameter("username"));
			a.setPassword(request.getParameter("password"));

			session.setAttribute("currentAdmin", a);
			userdao.updateUser(a);

			String view = null;
			if (result.hasErrors())
```

```
                        view = "admin-update";
                else {
                        status.isComplete();
                        view = "admin-dashboard";
                }
                return view;
        } else
                return "home";
    }

    // log out admin
    @GetMapping("/admin/logout.htm")
    public String logoutAdmin(HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession(false);
        if (session != null) {
                session.invalidate();
        }
        User user = new User();
        model.addAttribute("user", user);
        return "user-login";
    }
}
```

2. User Controller:

```
package com.me.wamp.controller;

import javax.servlet.http.HttpServletRequest;
//import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.support.SessionStatus;

import com.me.wamp.dao.ArtistDAO;
import com.me.wamp.dao.MediaDAO;
import com.me.wamp.dao.PlaylistDAO;
import com.me.wamp.dao.UserDAO;
import com.me.wamp.pojo.Media;
import com.me.wamp.pojo.Playlist;
import com.me.wamp.pojo.User;
import com.me.wamp.validation.UserValidator;

import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;

@Controller
public class UserController {

	@Autowired
	UserValidator userValidator;

	@GetMapping("/user/login.htm")
	public String showLoginForm(ModelMap model) {
		User user = new User();
		model.addAttribute("user", user);
		return "user-login";
	}

	@PostMapping("/user/login.htm")
	public String handleLoginForm(@ModelAttribute("user") User user,
BindingResult result, SessionStatus status,
```

```
                UserDAO userdao, HttpServletRequest request) {
            if (result.hasErrors()) {
                return "user-registration";
            } else {
                HttpSession session = request.getSession(true);
                User u = userdao.getUser(user);
                if (u != null) {
                    session.setAttribute("currentUser", u);
                    System.out.println(u.getPlaylist());
                    status.setComplete();
                    return "user-dashboard";
                } else {
                    request.setAttribute("error", "Incorrect Credentials");
                    return "user-login";
                }
            }
        }

        @GetMapping("/user/save.htm")
        public String saveUserGet(ModelMap model) {
            User user = new User(); // FormBackingObject
            model.addAttribute("user", user);
            return "user-registration";
        }

        @PostMapping("/user/save.htm")
        public String saveUserPost(@ModelAttribute("user") User user, BindingResult
result, SessionStatus status,
                UserDAO userdao, PlaylistDAO playlistdao, HttpServletRequest
request) {
            String view = null;
            if (result.hasErrors())
                view = "user-registration";
            else {
                User u = userdao.getUser(user);
                if (u == null) {
                    user.setRole("user");

                    Playlist playlist = new Playlist();
                    playlist.setName(user.getUsername() + "_playlist");
                    playlist.setType("song");

                    playlistdao.createPlaylist(playlist);

                    user.setPlaylist(playlist);

                    userdao.createUser(user);
```

```
                                sendEmail(request.getParameter("email"),
user.getUsername());

                                status.isComplete();
                                view = "user-registrationSuccess";
                        } else {
                                request.setAttribute("error", "User already exists");
                                view = "user-registration";
                        }
                }
                return view;
        }

        public void sendEmail(String email, String username) {
                String to = email;
                final String from = "wampmediaplayer@gmail.com";
                final String password = "Password@123";

                Properties props = new Properties();
                props.put("mail.smtp.host", "smtp.gmail.com");
                props.put("mail.smtp.socketFactory.port", "465");
                props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
                props.put("mail.smtp.auth", "true");
                props.put("mail.smtp.port", "465");
                // get Session
                Session session = Session.getDefaultInstance(props, new
javax.mail.Authenticator() {
                        protected PasswordAuthentication getPasswordAuthentication() {
                                return new PasswordAuthentication(from, password);
                        }
                });

                try {
                        MimeMessage message = new MimeMessage(session);
                        message.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));
                        message.setSubject("!WAMPED!");
                        message.setText("Hello " + username
                                        + ", <br><br>Welcome to the world of
WAMP.<br><br>You have successfully registered on WAMP, a simple Web Accessible
Media Player to cater to your Media needs.<br><br> <a style=\"background-color:
darkgreen; padding: 10px; color: white; font-family: 'Trebuchet MS', 'Lucida Sans
Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;\"
href=\"http://localhost:8080/wamp/user/login.htm\" target=\"_blank\">Let's
WAMP!</a><br><br>Thank You,<br>WAMP Team.",
```

```
                                        "UTF-8", "html");

                        // Send message
                        Transport.send(message);
                } catch (Exception mex) {
                        mex.printStackTrace();
                }
        }

        @GetMapping("/user/mediaSearch.htm")
        public String searchMediaGet(@ModelAttribute("user") User user, BindingResult
result, SessionStatus status,
                        UserDAO userdao, MediaDAO mediadao, HttpServletRequest
request) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentUser") != null) {
                        User u = (User) session.getAttribute("currentUser");
                        if (u.getRole().equals("user")) {
                                String medianame = request.getParameter("media-
keyword");
                                ArrayList<Media> mediaList = (ArrayList<Media>)
mediadao.getMedia(medianame);
                                if (mediaList.size() > 0) {
                                        request.setAttribute("mediaList", mediaList);
                                        return "user-media-list";
                                } else {
                                        request.setAttribute("error", "Media does not exist");
                                        return "user-dashboard";
                                }
                        }
                        return "user-dashboard";
                } else
                        return "home";
        }

        @GetMapping("/user/userSearch.htm")
        public String searchUserGet(@ModelAttribute("user") User user, BindingResult
result, SessionStatus status,
                        UserDAO userdao, HttpServletRequest request) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentUser") != null) {
                        String username = request.getParameter("user-keyword");
                        User searchUser = userdao.getUserByUsername(username);
                        if (searchUser != null) {
                                request.setAttribute("userList", searchUser);
                                return "user-user-list";
                        } else {
```

```
                        request.setAttribute("error", "User not found");
                        return "user-dashboard";
                    }
                } else
                    return "home";
            }

            @GetMapping("/user/userDashboard.htm")
            public String userDasboard(HttpServletRequest request) {
                HttpSession session = request.getSession(true);
                if (session.getAttribute("currentUser") != null) {
                    return "user-dashboard";
                } else
                    return "home";
            }

            @GetMapping("/user/logout.htm")
            public String logoutUser(HttpServletRequest request, ModelMap model) {
                HttpSession session = request.getSession(false);
                if (session != null) {
                    session.invalidate();
                }
                User user = new User();
                model.addAttribute("user", user);
                return "user-login";
            }

            @GetMapping("/user/delete.htm")
            public String deleteUser(UserDAO userdao, PlaylistDAO playlistdao,
HttpServletRequest request, ModelMap model) {
                HttpSession session = request.getSession(true);
                if (session != null) {
                    User user = (User) session.getAttribute("currentUser");
                    user = userdao.getUserById(user.getUserid());

            playlistdao.deletePlaylist(playlistdao.getPlaylist(user.getPlaylist().getPlaylistid()));
                    userdao.deleteUser(user);
                    session.invalidate();
                }
                User user = new User();
                model.addAttribute("user", user);
                return "user-login";
            }

            @GetMapping("/user/playlist.htm")
            public String userPlaylist(HttpServletRequest request, PlaylistDAO playlistdao) {
                HttpSession session = request.getSession(true);
```

```
		if (session.getAttribute("currentUser") != null) {
			return "user-playlist";
		} else
			return "home";
	}

	@GetMapping("/user/addMedia.htm")
	public String userAddMediaPlaylist(MediaDAO mediadao, UserDAO userdao,
PlaylistDAO playlistdao,
			HttpServletRequest request) {
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentUser") != null) {
			int mediaid = Integer.parseInt(request.getParameter("mediaid"));
			Media media = mediadao.getMediaById(mediaid);
			Playlist playlist = ((User)
session.getAttribute("currentUser")).getPlaylist();
			playlist.addMedia(media);
			((User) session.getAttribute("currentUser")).setPlaylist(playlist);
			playlistdao.updatePlaylist(playlist);
			userdao.updateUser(((User) session.getAttribute("currentUser")));
			return "user-playlist";
		} else
			return "home";
	}

	@GetMapping("/user/deleteMedia.htm")
	public String userDeleteMediaPlaylist(MediaDAO mediadao, UserDAO userdao,
PlaylistDAO playlistdao,
			HttpServletRequest request) {
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentUser") != null) {
			int mediaid = Integer.parseInt(request.getParameter("mediaid"));
			Media media = mediadao.getMediaById(mediaid);
			Playlist playlist = ((User)
session.getAttribute("currentUser")).getPlaylist();
			System.out.println(playlist.getMedialist().size());
			playlist.removeMediaById(mediaid);
			((User) session.getAttribute("currentUser")).setPlaylist(playlist);
			playlistdao.updatePlaylist(playlist);
			userdao.updateUser((User) session.getAttribute("currentUser"));
			return "user-playlist";
		} else
			return "home";
	}

	@GetMapping("/user/update.htm")
	public String updateUserGet(HttpServletRequest request) {
```

```
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentUser") != null) {
			return "user-update";
		} else
			return "home";
	}

	@PostMapping("/user/update.htm")
	public String updateUserPost(@ModelAttribute("user") User user, BindingResult
result, SessionStatus status,
			UserDAO userdao, HttpServletRequest request) {
		HttpSession session = request.getSession(true);
		if (session.getAttribute("currentUser") != null) {
			User u = (User) session.getAttribute("currentUser");
			u.setUsername(request.getParameter("username"));
			u.setPassword(request.getParameter("password"));
			session.removeAttribute("currentUser");
			session.setAttribute("currentUser", u);
			userdao.updateUser(u);

			String view = null;
			if (result.hasErrors())
				view = "user-update";
			else {
				status.isComplete();
				view = "user-dashboard";
			}
			return view;
		} else
			return "home";
	}

}
```

3. Media Controller:

```
package com.me.wamp.controller;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
//import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.support.SessionStatus;

import com.me.wamp.dao.FileDAO;
import com.me.wamp.dao.MediaDAO;
import com.me.wamp.pojo.Media;
import com.me.wamp.validation.MediaValidator;

@Controller
public class MediaController {

    @Autowired
    MediaValidator mediaValidator;

    @GetMapping("/media/browse.htm")
    public String browseMedia(MediaDAO mediadao, HttpServletRequest request) {
        ArrayList<Media> mediaList = (ArrayList<Media>) mediadao.getAllMedia();
        request.setAttribute("mediaList", mediaList);
        return "media-browseList";
    }

    @GetMapping("/media/save.htm")
    public String saveMediaGet(ModelMap model) {
        Media media = new Media();
        model.addAttribute("media", media);
        return "media-add";
    }

    @PostMapping("/media/save.htm")
    public String saveMediaPost(@ModelAttribute("media") Media media, BindingResult
result, SessionStatus status, MediaDAO mediadao) {
        mediadao.createMedia(media);

        String view=null;
        if (result.hasErrors())
            view = "media-add";
        else {
            status.isComplete();
            view = "media-addSuccess";
        }
```

```
        return view;
    }

    @GetMapping("/media/search.htm")
    public String searchMediaGet(MediaDAO mediadao, HttpServletRequest request) {
            String media = request.getParameter("search");

            ArrayList<Media> mediaList = (ArrayList<Media>) mediadao.getMedia(media);

            request.setAttribute("mediaList", mediaList);
            return "media-list";
    }

    @GetMapping("/media/delete.htm")
    public String deleteMedia(MediaDAO mediadao, HttpServletRequest request) {

            String mediaid = request.getParameter("deleteMedia");
            Media media = mediadao.getMediaById(Integer.parseInt(mediaid));
            mediadao.deleteMedia(media);

            HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }
        return "media-browseList";
    }

    @GetMapping("/media/play.htm")
    public String playMedia(MediaDAO mediadao, FileDAO filedao, HttpServletRequest
request) {

            String mediaid = request.getParameter("mediaid");

            Media media = mediadao.getMediaById(Integer.parseInt(mediaid));

            request.setAttribute("name", media.getName());

            if(media.getUrl().matches("https://www.youtube.com/(.*)")) {
                    request.setAttribute("url",
"https://www.youtube.com/embed/"+media.getUrl().split("=")[1]+"?rel=0&showinfo=0&aut
ohide=1&modestbranding=1&fs=0");
                    request.setAttribute("location", "web");
            }
            else {
                    try {
                            File file = new
File("./resources/videos/"+filedao.getFile(Integer.parseInt(media.getUrl())).getName());
```

```
                        OutputStream os = new FileOutputStream(file);

        os.write(filedao.getFile(Integer.parseInt(media.getUrl())).getData());
                        os.close();
                        request.setAttribute("url",
"http://localhost:8080/wamp/resources/videos/"+filedao.getFile(Integer.parseInt(media.g
etUrl())).getName());
                request.setAttribute("location", "db");
                } catch (Exception fe) {
                        fe.printStackTrace();
                }
        }

        if(media.getType().equals("Song")||media.getType().equals("song")) {
                request.setAttribute("type", "song");
        } else {
                request.setAttribute("type", "video");
        }

        return "media-play";
   }

   @PostMapping("/media/update.htm")
   public String updateMediaPost(@ModelAttribute("media") Media media,
BindingResult result, SessionStatus status, MediaDAO mediadao) {
      mediadao.updateMedia(media);

      String view=null;
      if (result.hasErrors())
         view = "media-update";
      else {
         status.isComplete();
         view = "media-browseList";
      }
      return view;
   }

}
```