```
import java.io.*;
import java.util.List;
import java.util.Map;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.zip.ZipOutputStream;


// Part 1: User Management

class User {
    private String username;
    private String password;
    private UserType userType;

    public User(String username, String password, UserType userType) {
        this.username = username;
        this.password = password;
        this.userType = userType;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public UserType getUserType() {
        return userType;
    }

    public void setUsername(String newUsername) {
        this.username = newUsername;
    }

    public void setPassword(String newPassword) {
        this.password = newPassword;
    }
}

enum UserType {
    EXTERNAL, ADMINISTRATOR
}

class UserManager {
    private List<User> users;

    public UserManager() {
        users = new ArrayList<>();
    }

    public void registerUser(User user) {
        users.add(user);
    }

    public User authenticateUser(String username, String password) {
        for (User user : users) {
            if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
                return user;
            }
        }
        return null;
    }

    public boolean authorizeUser(User user, Privilege privilege) {
        if (user.getUserType() == UserType.ADMINISTRATOR) {
            return true;
        } else {
            return false;
        }
    }

    public void manageProfile(User user, String newUsername, String newPassword) {
        user.setUsername(newUsername);
        user.setPassword(newPassword);
    }
}

class Privilege {
    // Define your Privilege class here
}
```

```
// Part 2: Metadata Management

class Metadata {
    private String projectName;
    private String projectVersion;
    private String projectDescription;
    private List<String> authors;
    private String contactDetails;

    public Metadata(String projectName, String projectVersion, String projectDescription,
List<String> authors, String contactDetails) {
        this.projectName = projectName;
        this.projectVersion = projectVersion;
        this.projectDescription = projectDescription;
        this.authors = authors;
        this.contactDetails = contactDetails;
    }

    public String getProjectName() {
        return projectName;
    }

    public String getProjectVersion() {
        return projectVersion;
    }

    public String getProjectDescription() {
        return projectDescription;
    }

    public List<String> getAuthors() {
        return authors;
    }

    public String getContactDetails() {
        return contactDetails;
    }
}

class MetadataManager {
    private Metadata metadata;

    public MetadataManager() {
        metadata = new Metadata("Electric Vehicle Charging Station", "1.0", "A system for
managing electric vehicle charging stations", Arrays.asList("Alice Smith", "Bob Jones"),
"contact@example.com");
  }

  public Metadata getMetadata() {
    return metadata;
  }

  public void updateMetadata(Metadata newMetadata) {
    metadata = newMetadata;
  }
}

// Part 3: Log File Management

enum LogType {
  SYSTEM, CHARGING_STATION, ENERGY_MANAGEMENT
}

class LogEntry {
  private String timestamp;
  private LogType logType;
  private String message;

  public LogEntry(String timestamp, LogType logType, String message) {
    this.timestamp = timestamp;
    this.logType = logType;
    this.message = message;
  }

  public String getTimestamp() {
    return timestamp;
  }

  public LogType getLogType() {
    return logType;
  }

  public String getMessage() {
    return message;
  }
}

class LogFileManager {
```

```
private Map<LogType, List<LogEntry>> logFiles;

public LogFileManager() {
   logFiles = new HashMap<>();
   for (LogType logType : LogType.values()) {
      logFiles.put(logType, new ArrayList<>());
   }
}

public void createLogEntry(LogType logType, String message) {
   String timestamp = getCurrentTimestamp();
   LogEntry logEntry = new LogEntry(timestamp, logType, message);
   logFiles.get(logType).add(logEntry);
}

public void moveLogFile(LogType logType, OutputStream destinationPath) {
   List<LogEntry> logEntries = logFiles.remove(logType);
   saveLogEntriesToFile(logEntries, destinationPath);
}

public void deleteLogFile(LogType logType) {
   logFiles.remove(logType);
}

public void archiveLogFile(LogType logType, String archivePath) {
   List<LogEntry> logEntries = logFiles.remove(logType);
   ZipOutputStream zipOutputStream = null;
   try {
      zipOutputStream = new ZipOutputStream(new FileOutputStream(archivePath));
   } catch (FileNotFoundException e) {
      throw new RuntimeException(e);
   }
   saveLogEntriesToFile(logEntries, zipOutputStream);
   try {
      zipOutputStream.closeEntry();
      zipOutputStream.close();
   } catch (IOException e) {
      e.printStackTrace();
   }
}

private String getCurrentTimestamp() {
   return new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
}
```

```
    private void saveLogEntriesToFile(List<LogEntry> logEntries, OutputStream outputStream) {
        try (BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(outputStream))) {
            for (LogEntry logEntry : logEntries) {
                writer.write(logEntry.getTimestamp() + " " + logEntry.getLogType() + " " +
logEntry.getMessage());
                writer.newLine();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

// Part 4: Data Exchange Simulation

class DataStream {
    private byte[] data;

    public DataStream(byte[] data) {
        this.data = data;
    }

    public byte[] getData() {
        return data;
    }

    public void sendData(DataReceiver receiver) {
        receiver.receiveData(data);
    }
}

class DataReceiver {
    public void receiveData(byte[] data) {
        // Process received data
    }
}

class DataStreamManager {
    public void simulateDataExchange(DataStream sender, DataReceiver receiver) {
        sender.sendData(receiver);
    }
}
```