BudgetMapper

```
package DataLayer;

import Domain.Budget;

import java.sql.*;
import java.util.ArrayList;

public class BudgetMapper {

    public boolean addBudget(int year, int quarter, int budget, Connection con) {

        String SQL = "insert into budget values(?, ?, ?, ?, ?)";

        PreparedStatement statement = null;

        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, budget);
            statement.setInt(2, year);
            statement.setInt(3, quarter);
            statement.setInt(4, 0);
            statement.setInt(5, 0);
            statement.executeUpdate();

        } catch(Exception e) {
            return false;
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return true;
    }

    public boolean modifyBudget(int new_budget, int year, int quarter, Connection con) {
        String SQL = "update budget set initial_budget=? where yearnum = ? and quarternum = ?";
        PreparedStatement statement = null;

        try {
        statement = con.prepareStatement(SQL);
            statement.setInt(1, new_budget);
            statement.setInt(2, year);
            statement.setInt(3, quarter);
            statement.executeUpdate();

        } catch(Exception e) {
            return false;
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return true;
    }

    public Budget getActiveBudget(int year, int quarter, Connection con) {
        String SQL = "select * from budget where yearnum = ? and quarternum = ?";
        PreparedStatement statement = null;
        ResultSet rs = null;
        Budget Budget = null;

        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, year);
            statement.setInt(2, quarter);
            rs = statement.executeQuery();


            while (rs.next()) {
                Budget = new Budget(
                        rs.getInt(1),
                        rs.getInt(2),
                        rs.getInt(3),
                        rs.getInt(5),
                        rs.getInt(4)
                );
            }

        } catch (Exception e) {
            return null;
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return Budget;
    }

    public ArrayList<Budget> getAllBudgets(Connection con) {
        String SQL = "select * from budget";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<Budget> BudgetCollection = new ArrayList<>();

        try {
            statement = con.prepareStatement(SQL);

            rs = statement.executeQuery();

            while (rs.next()) {
                BudgetCollection.add(new Budget(
                        rs.getInt(1),
                        rs.getInt(2),
                        rs.getInt(3),
                        rs.getInt(4),
                        rs.getInt(5)
                ));
            }

        } catch (Exception e) {
            System.out.println("error in BudgetMapper");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return BudgetCollection;
    }

    public int getAvailableFunds(int year, int quarter, Connection con) {
        int availablefunds = 0;
        PreparedStatement statement = null;
        String SQL = "select initial_budget, reserved from budget where yearnum=? and quarternum = ?";

        ResultSet rs = null;
        try {
```

```java
            statement = con.prepareStatement(SQL);
            statement.setInt(1, year);
            statement.setInt(2, quarter);

            rs = statement.executeQuery();

            while (rs.next()) {
                availablefunds = rs.getInt(1) - rs.getInt(2);
            }

        } catch (Exception e) {
            System.out.println("Error in availableFunds");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return availablefunds;
    }

}
```

CompanyMapper

```java
package DataLayer;

import Domain.Company;

import java.sql.*;
import java.util.ArrayList;

public class CompanyMapper {

    public Company getCompanyById(int id, Connection con) {
        Company company = null;
        String SQL = "select * from companies where id= ? ";

        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, id);
            rs = statement.executeQuery();
            if (rs.next()) {
                company = new Company(rs.getInt(1),
                        rs.getString(2),
                        rs.getString(3),
                        rs.getString(4));
            }

        } catch (Exception e) {
            System.out.println("Error in CompanyMapper");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return company;
    }

    public int createCompany(String company_name, String country_code, Connection con) {
        String SQL = "insert into companies values (?,?,?,?)";
    PreparedStatement statement = null;
    try {
        statement = con.prepareStatement(SQL);

        int nextCompanyId = getNextCompanyId(con);

        statement.setInt(1, nextCompanyId);
        statement.setString(2, company_name);
        statement.setString(3, null);
        statement.setString(4, country_code);

        statement.executeUpdate();

        return nextCompanyId;

    } catch (Exception e) {
        System.out.println("Error in createCompany");
    } finally {
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return -1;
}

public void updateCompanyLogo(String filename, int id, Connection con) {
    String SQL = "update companies set IMAGE_FILENAME=? where id=?";

    PreparedStatement statement = null;
    try {
        statement = con.prepareStatement(SQL);
        statement.setString(1, filename);
        statement.setInt(2, id);
        statement.executeUpdate();
    } catch (Exception e) {
        System.out.println("Error in updateCompanyLogo");
    } finally {
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
}

 public int getNextCompanyId(Connection con) {
     PreparedStatement statement = null;
     ResultSet rs = null;

     String SQL = "select MAX(id) from companies";
     int id = 0;

    try {
        statement = con.prepareStatement(SQL);
        rs = statement.executeQuery();
        while (rs.next()) {
            id = rs.getInt(1);
        }

    } catch (Exception e) {
        System.out.println("Error in CompanyMapper - getNextCompanyId()");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return id + 1;
}
```

```java
public ArrayList getCompanies(Connection con) {
    ArrayList<Company> companies = new ArrayList<>();
    String SQL = "select * from companies";

    PreparedStatement statement = null;
    ResultSet rs = null;
    try {
        statement = con.prepareStatement(SQL);
        rs = statement.executeQuery();
        while(rs.next()) {
            companies.add(new Company(rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4)));
        }

    } catch (Exception e) {
        System.out.println("Error in CompanyMapper");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return companies;
}
public ArrayList getCompanyNames(String query, int company_id, Connection con) {
    ArrayList<String> results = new ArrayList<>();
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select DISTINCT name\n" +
            "from companies\n" +
            "where lower(name) like lower('%" + query + "%')\n";

    System.out.println(SQL);

    try {
        statement = con.prepareStatement(SQL);
        rs = statement.executeQuery();
        while (rs.next()) {
            results.add(rs.getString(1));
        }
    } catch (Exception e) {
        System.out.println("Error in ProjectMapper - getCompanyNames()");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return results;
}

public int getCompanyIdByName(String name) {
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) { }

    System.out.println("name getcompanyidbyname: " + name);

    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select id " +
            "from companies " +
            "where name=?";
        int id = -1;

        try {
            statement = con.prepareStatement(SQL);
            statement.setString(1, name);
            rs = statement.executeQuery();
            if (rs.next()) {
                id = rs.getInt(1);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            System.out.println("Error in CompanyMapper - getCompanyIdByName()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id;
    }

}
```

DatabaseConnection

```java
package DataLayer;

import com.mchange.v2.c3p0.ComboPooledDataSource;


import java.beans.PropertyVetoException;
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;

public class DatabaseConnection {

    private ComboPooledDataSource cpds;
    private static DatabaseConnection datasource;

    private DatabaseConnection() throws IOException, SQLException,PropertyVetoException {

        cpds = new ComboPooledDataSource();
        cpds.setDriverClass("oracle.jdbc.driver.OracleDriver"); //loads the jdbc driver
        cpds.setJdbcUrl("jdbc:oracle:thin:@datdb.cphbusiness.dk:1521:dat");
        cpds.setUser("cphlj222");
        cpds.setPassword("cphlj222");

        cpds.setInitialPoolSize(7);
        cpds.setAcquireIncrement(15);
        cpds.setMaxPoolSize(1000);
        cpds.setMinPoolSize(3);
        cpds.setMaxStatements(10);
        cpds.setIdleConnectionTestPeriod(2000);
    }

    public static DatabaseConnection getInstance() throws IOException, SQLException,PropertyVetoException {

        if(datasource == null){
            synchronized (DatabaseConnection.class) {
                if(datasource == null){
                    datasource = new DatabaseConnection();
                }
            }
        }
        return datasource;
    }

    public Connection getConnection() throws SQLException {
        return this.cpds.getConnection();
    }

}
```

DatabaseFacade

```
package DataLayer;

import java.sql.Connection;
import java.sql.Timestamp;
import java.util.ArrayList;

import Domain.*;

public class DatabaseFacade {

    private Connection getCon() {
        try {
            return DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {};
        return null;
    }

    public User getUserById(int user_id) { return new UserMapper().getUserById(user_id, getCon()); }

    // PROJECT
    public int createProjectRequest(int budget, String project_body, User user, String project_type, Timestamp
execution_date) {
        return new ProjectMapper().createProjectRequest(budget, project_body, user, project_type, execution_date, getCon());

    }
    public ArrayList<Project> getProjectsByCompanyId(int company_id) {
        return new ProjectMapper().getProjectsByCompanyId(company_id, getCon());
    }
    public ArrayList<Project> getProjectsByUserId(int user_id) {
        return new ProjectMapper().getProjectsByCompanyId(user_id, getCon());
    }

    public boolean changeProjectStatus(int project_id, String new_status, int companyId, int userId) {
        return new ProjectMapper().changeProjectStatus(project_id, new_status, companyId, userId, getCon());
    }

    public DisplayProject getProjectById(int id, int companyId) {
        return new ProjectMapper().getProjectById(id, companyId, getCon());
    }
    public ArrayList getMessagesByProjectId(int projId) { return new MessageMapper().getMessagesByProjectId(projId,
getCon()); }
    public Message postMessage(int userId, int projId, String body, int companyId) { return new
MessageMapper().postMessage(userId, projId, body, companyId, getCon());}
    public void markRead(int id, int companyId) {
        new ProjectMapper().changeReadStatus(0, id, companyId, getCon());
    }
    public ArrayList getProjectsByState(String state, int companyId) {
        return new ProjectMapper().getProjectsByState(state, companyId, getCon());
    }
    public ArrayList getProjectsByType(String type, int companyId) {
        return new ProjectMapper().getProjectsByType(type, companyId, getCon());
    }
    public ArrayList getProjectsByCompanyName(String companyName, int companyId) {
        return new ProjectMapper().getProjectsByCompanyName(companyName, companyId, getCon());
```

```java
    }

    //Search
    public ArrayList search(String q, int companyId) {
        return new ProjectMapper().search(q, companyId, getCon());
    }

    public int[] getStatusCounts(int companyId) {
        return new ProjectMapper().getStatusCounts(companyId, getCon());
    }
    public ArrayList getStagesByProjectId(int project_id) {
        return  new ProjectMapper().getStagesByProjectId(project_id, getCon());
    }
    public void updateChangeDate(int project_id, int company_id) {
        new ProjectMapper().updateChangeDate(project_id, company_id);
    }
    public void updateNotification(int project_id, String notification) {
        new ProjectMapper().updateNotification(project_id, notification);
    }
    public void markUnread(int project_id, int company_id) {
        new ProjectMapper().changeReadStatus(1, project_id, company_id, getCon());
    }

    public ArrayList getDistinctStatuses(String query) {
        return new ProjectMapper().getDistinctStatuses(query, getCon());
    }
    public ArrayList getDistinctTypes(String query, int companyId) {
        return new ProjectMapper().getDistinctTypes(query, companyId, getCon());
    }

    // COMPANY
    public Company getCompanyById(int id) {
        return new CompanyMapper().getCompanyById(id, getCon());
    }

    public int createCompany(String company_name, String country_code) { return new
CompanyMapper().createCompany(company_name, country_code, getCon()); };

    public void updateCompanyLogo(String filename, int id) { new CompanyMapper().updateCompanyLogo(filename, id,
getCon()); }

    public ArrayList<Company> getCompanies() { return new CompanyMapper().getCompanies(getCon()); }
    public ArrayList<Company> getCompanyNames(String query, int companyId) { return new
CompanyMapper().getCompanyNames(query, companyId, getCon()); }
    public int getCompanyIdByName(String name) {return new CompanyMapper().getCompanyIdByName(name);}

    // USER
    public User getUserByEmail(String email) {
        return new UserMapper().getUserByEmail(email, getCon());
    }
    public int createUser(String name,String user_email, int company_id) {
        return new UserMapper().createUser(name, user_email, company_id, getCon());
    }
    public ArrayList<User> getUsersByCompanyId(int company_id) {
        return new UserMapper().getUsersByCompanyId(company_id, getCon());
    }
    public ArrayList getUserInfoInvolvedInProjectById(int project_id, int user_id) { return new
UserMapper().getUserInfoInvolvedInProjectById(project_id, user_id, getCon()); }

    public ArrayList<User> getUsers() {
        return new UserMapper().getUsers(getCon());
    }

    public boolean markUserDeleted(int user_id) {
        return new UserMapper().markUserDeleted(user_id, getCon());
    }
    // POE
    public boolean addPoeFile(int project_id, String filename, int user_id, String filetype, int stage) {
        return new PoeMapper().addPoeFile(project_id, filename, user_id, filetype, stage, getCon());
    }

    public ArrayList<Poe> getPoe(int project_id) {
        return new PoeMapper().getPoe(project_id, getCon());
    }

    public boolean deletePoe(int fileId) {
        return new PoeMapper().deletePoe(fileId, getCon());
    }
    public boolean markDeletePoe(int fileId) {
        return new PoeMapper().markDeletePoe(fileId, getCon());
    }

    // BUDGET
    public boolean addBudget(int year, int quarter, int budget) {
    return new BudgetMapper().addBudget(year, quarter, budget, getCon());
    }

    public boolean modifyBudget(int new_budget, int year, int quarter) {
    return new BudgetMapper().modifyBudget(new_budget, year, quarter, getCon());
    }

    public Budget getActiveBudget(int year, int quarter) {
    return new BudgetMapper().getActiveBudget(year, quarter, getCon());
    }

    public ArrayList<Budget> getAllBudgets() {
        return new BudgetMapper().getAllBudgets(getCon());
    }

    public int getAvailableFunds(int year, int quarter) {
        return new BudgetMapper().getAvailableFunds(year, quarter, getCon());
    }

    // Nonce
    public int addNonce(Nonce nonce) { return new NonceMapper().addNonce(nonce, getCon());}
    public int getUserIdByNonce(String nonce) { return new NonceMapper().getUserIdByNonce(nonce, getCon());}
    public boolean createPassword(int id, String password) { return new UserMapper().createPassword(id, password,
getCon()); }
    public void deleteNonce(String nonce) { new NonceMapper().deleteNonce(nonce, getCon()); }


    // Statistics
    public ArrayList<String[]> getDistinctTypesCounts              (Timestamp start, Timestamp end) { return
StatisticsGetter.getDistinctTypesCounts(start, end, getCon()); }
    public ArrayList<String[]> getAvgCostOfProjectsByCountry       (Timestamp start, Timestamp end) { return
StatisticsGetter.getAvgCostOfProjectsByCountry(start, end, getCon()); }
    public ArrayList<String[]> getAvgCostPerType                   (Timestamp start, Timestamp end) { return
StatisticsGetter.getAvgCostPerType(start, end, getCon()); }
    public ArrayList<String[]> getAvgMessagesPerProject            (Timestamp start, Timestamp end) { return
StatisticsGetter.getAvgMessagesPerProject(start, end, getCon()); }
    public ArrayList<String[]> getBudgetProgression                (Timestamp start, Timestamp end) { return
StatisticsGetter.getBudgetProgression(start, end, getCon()); }
    public ArrayList<String[]> getCompaniesByLargestApprovedBudget  (Timestamp start, Timestamp end) { return
StatisticsGetter.getCompaniesByLargestApprovedBudget(start, end, getCon()); }
    public ArrayList<String[]> getCostPerType                      (Timestamp start, Timestamp end) { return
StatisticsGetter.getCostPerType(start, end, getCon()); }
    public ArrayList<String[]> getCountOfFinishedProjects          (Timestamp start, Timestamp end) { return
StatisticsGetter.getCountOfFinishedProjects(start, end, getCon()); }
    public ArrayList<String[]> getCountOfMessages                  (Timestamp start, Timestamp end) { return
StatisticsGetter.getCountOfMessages(start, end, getCon()); }
    public ArrayList<String[]> getCountOfProjectsByCountry         (Timestamp start, Timestamp end) { return
StatisticsGetter.getCountOfProjectsByCountry(start, end, getCon()); }
    public ArrayList<String[]> getMoneyReimbursed                  (Timestamp start, Timestamp end) { return
StatisticsGetter.getMoneyReimbursed(start, end, getCon()); }
    public ArrayList<String[]> getTypesWithHighestSuccessRate       (Timestamp start, Timestamp end) { return
StatisticsGetter.getTypesWithHighestSuccessRate(start, end, getCon()); }
}
```

MessageMapper

```
package DataLayer;

import Domain.Message;

import java.sql.*;
import java.util.ArrayList;
public class MessageMapper {

    public ArrayList getMessagesByProjectId(int id, Connection con) {
        ArrayList<Message> messages = new ArrayList<>();
        String SQL = "select * from messages where project_id =" + id;

        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            statement = con.prepareStatement(SQL);

            rs = statement.executeQuery();
            while (rs.next()) {
                messages.add(new Message(rs.getInt(1),
                        rs.getInt(2),
                        rs.getInt(3),
                        rs.getString(4),
                        rs.getTimestamp(5)
                ));
            }

        } catch (Exception e) {
            System.out.println("Error in MessageMapper.getMessagesByProjectId");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return messages;
    }

    public Message getMessageById(int id, Connection con) {
        Message message = null;
        String SQL = "select * from messages where id =" + id;

        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            statement = con.prepareStatement(SQL);

            rs = statement.executeQuery();
            if (rs.next()) {
                message = new Message(rs.getInt(1),
                        rs.getInt(2),
                        rs.getInt(3),
                        rs.getString(4),
                        rs.getTimestamp(5)
                );
            }

        } catch (Exception e) {
            System.out.println("Error in MessageMapper.getMessagesByProjectId");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return message;
    }

    public Message postMessage(int userId, int projectId, String body, int companyId, Connection con) {

        String SQL = "insert into messages values (?,?,?,?,?)";

        PreparedStatement statement = null;
        int nextMessageId = 0;
        try {
            statement = con.prepareStatement(SQL);
            nextMessageId = getNextMessageId(con);

            java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
            Timestamp timestamp = new Timestamp(date.getTime());

            statement.setInt(1, nextMessageId);
            statement.setInt(2, userId);
            statement.setInt(3, projectId);
            statement.setString(4, body);
            statement.setTimestamp(5, timestamp);
            statement.executeUpdate();

            DatabaseFacade facade = new DatabaseFacade();
            facade.updateChangeDate(projectId, companyId);
            facade.updateNotification(projectId, "New message!");
            if(companyId == 1)
                facade.markUnread(projectId, 2); // 2 is just not dell, a la partner
            else
                facade.markUnread(projectId, 1);

        } catch (SQLException t) {
            System.out.println("SQLException in Messagemapper() - postMessage*(");
        }
        catch (Exception e) {
            System.out.println("Error in Messagemapper - postMessage()");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        try {
              con = DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {};
        return getMessageById(nextMessageId, con);
    }

    public int getNextMessageId(Connection con) {
        PreparedStatement statement = null;
        ResultSet rs = null;
        String SQL = "select MAX(id) from messages";
        int id = 0;
        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
            }

        } catch (Exception e) {
            System.out.println("Error in MessageMapper - getNextMessageId()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id + 1;
    }
}
```

NonceMapper

```
package DataLayer;

import Domain.Nonce;
import Domain.Stage;

import java.sql.*;
import java.util.ArrayList;

public class NonceMapper {

    public int addNonce(Nonce nonce, Connection con) {
        PreparedStatement statement = null;
        java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
        Timestamp timestamp = new Timestamp(date.getTime());
        int id = getNextNonceId();
        String SQL = "insert into nonces values(?,?,?,?,?)";

        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, id);
            statement.setString(2, nonce.getNonce());
            statement.setInt(3, nonce.associate_id);
            statement.setTimestamp(4, timestamp);
            statement.setString(5, nonce.getType());

            statement.executeUpdate();

            return id;
        } catch (Exception e) {
            System.out.println("Exception in addNonce()");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return -1;

    }

    public int getUserIdByNonce(String nonce, Connection con) {
        PreparedStatement statement = null;
        ResultSet rs = null;
        String SQL = "select associate_id from nonces where nonce=?";

        int id = 0;

        try {
            statement = con.prepareStatement(SQL);
            statement.setString(1, nonce);
            rs = statement.executeQuery();
            if(rs.next()) {
                id = rs.getInt(1);
            }
        } catch (Exception e) {
            System.out.println("Error in NonceMapper - getUserIdByNonce()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id;
    }

    public int getNextNonceId() {
        PreparedStatement statement = null;
        ResultSet rs = null;
        Connection con = null;
        try {
            con = DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {};
        String SQL = "select MAX(id) from nonces";
        int id = 0;

        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
            }

        } catch (Exception e) {
            System.out.println("Error in NonceMapper - getNextNonceId()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id + 1;
    }


    public void deleteNonce(String nonce, Connection con) {
        PreparedStatement statement = null;
        String SQL = "delete from nonces" +
                " where nonce=?";

        try {
            statement = con.prepareStatement(SQL);
            statement.setString(1, nonce);
            statement.executeUpdate();

        } catch (Exception e) {
            System.out.println("Error in NonceMapper - deleteNonce()");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

    }
}
```

PoeMapper

```
package DataLayer;
```

```java
import Domain.Company;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import Domain.Poe;


/**
 * Created by Lasse on 17-04-2015.
 */

import javax.xml.crypto.Data;
import javax.xml.transform.Result;
import java.security.interfaces.RSAKey;
import java.sql.*;
import java.util.ArrayList;

public class PoeMapper {

    public boolean addPoeFile(int project_id, String filename, int user_id, String filetype, int stage, Connection con) {

        String SQL = "insert into poes values(?, ?, ?, ?, ?, ?, ?, ?)";

        PreparedStatement statement = null;

        java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
        Timestamp timestamp = new Timestamp(date.getTime());

        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, getNextPoEId());
            statement.setInt(2, project_id);
            statement.setString(3, filename);
            statement.setInt(4, user_id);
            statement.setTimestamp(5, timestamp);
            statement.setString(6, filetype);
            statement.setTimestamp(7, null);
            statement.setInt(8, stage);

            statement.executeUpdate();


        } catch(Exception e) {
            return false;
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return true;
    }

    public boolean deletePoe(int fileId, Connection con) {

        String SQL = "delete from poes where id=?";
        PreparedStatement statement = null;

        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, fileId);

            statement.executeUpdate();

        } catch (Exception e) {
            System.out.println("couldn't delete poe from db");
            return false;
        }finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return true;

    }

    public boolean markDeletePoe(int fileId, Connection con) {
        String SQL = "update poes set deletion_date=? where id=?";
        PreparedStatement statement = null;

        java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
        Timestamp timestamp = new Timestamp(date.getTime());

        try {
            statement = con.prepareStatement(SQL);

            statement.setTimestamp(1, timestamp);
            statement.setInt(2, fileId);

            statement.executeUpdate();

        } catch (Exception e) {
            System.out.println("couldn't delete poe from db");
            return false;
        }finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return true;

    }

    public ArrayList<Poe> getPoe(int project_id, Connection con) {

        String SQL = "select * from poes where project_id = ? order by creation_date";
        //int amountPoes = getAmountPoesByProjectId(project_id);
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<Poe> PoeCollection = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, project_id);

            rs = statement.executeQuery();


            while (rs.next()) {
                PoeCollection.add(new Poe(rs.getInt(1),
                        rs.getInt(2),
                        rs.getString(3),
                        rs.getInt(4),
                        rs.getTimestamp(5),
                        rs.getString(6),
                        rs.getTimestamp(7),
                        rs.getInt(8)
                ));
            }
        } catch (Exception e) {
            System.out.println("error in fisk");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return PoeCollection;

    }

    public int getNextPoEId() {
        PreparedStatement statement = null;
        ResultSet rs = null;
        Connection con = null;
        try {
            con = DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {};
        String SQL = "select MAX(id) from poes";
        int id = 0;

        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
            }

        } catch (Exception e) {
            System.out.println("Error in PoeMapper - getNextPoEId()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id + 1;
    }

}
```

PoeMapperTest

```
package DataLayer;

import Domain.Poe;
import org.junit.After;
import org.junit.Before;

import java.util.ArrayList;

import static org.junit.Assert.*;

public class PoeMapperTest {

    PoeMapper mapper;
    int file_id;
    int proj_id;
    String filename;
    int user_id;
    String filetype;
    int stage_id;
    @Before
    public void setUp() throws Exception {
        mapper = new PoeMapper();

    }

    @org.junit.Test
    public void testAddPoeFile() throws Exception {
        System.out.println("DUMMY VALUES:");
        stage_id = 2000;
        file_id = 100;
        proj_id = 10;
        filename = "eyyy.txt";
        user_id = 2;
        filetype = "txt";


        mapper.addPoeFile(proj_id, filename, user_id, filetype, stage_id, DatabaseConnection.getInstance().getConnection());

        ArrayList<Poe> poes= mapper.getPoe(proj_id, DatabaseConnection.getInstance().getConnection());

        int checkForMatch = -1;

        for (int i = 0; i < poes.size(); i++) {
            if (filename.equals(poes.get(i).getFilename())) {
                checkForMatch = i;
            }
        }
        if (checkForMatch != -1) {
            System.out.println(filename);
            System.out.println(poes.get(checkForMatch).getFilename());
            assertTrue(poes.get(checkForMatch).getFilename().equals(filename));
        } else {
            fail();
        }
    }

    @After
    public void resetDbChanges() throws Exception {

        mapper.deletePoe(file_id, DatabaseConnection.getInstance().getConnection());
    }

    @org.junit.Test
    public void testDeletePoe() throws Exception {

    }

    @org.junit.Test
    public void testGetPoe() throws Exception {

    }

    @org.junit.Test
    public void testGetNextPoEId() throws Exception {

    }
}
```

ProjectMapper

```
package DataLayer;

import Domain.DisplayProject;
```

```java
import Domain.Project;
import Domain.Stage;
import Domain.User;
import Domain.Result;

import javax.xml.crypto.Data;
import java.security.interfaces.RSAKey;
import java.sql.*;
import java.util.ArrayList;
import java.util.Calendar;


public class ProjectMapper {


    public int createProjectRequest(int budget, String project_body, User user, String project_type, Timestamp
execution_date, Connection con) {

        String SQL = "insert into projects values (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

        PreparedStatement statement = null;

        try {
            int nextProjectID = getNextProjectId();
            statement = con.prepareStatement(SQL);

            java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
            Timestamp timestamp = new Timestamp(date.getTime());


            statement.setInt(1, nextProjectID);
            statement.setTimestamp(2, timestamp);
            statement.setTimestamp(3, null);
            statement.setInt(4, user.company_id);
            statement.setInt(5, user.id);
            statement.setString(6, "Waiting Project Verification");
            statement.setInt(7, budget);
            statement.setString(8, project_body);
            statement.setTimestamp(9, execution_date);
            statement.setTimestamp(10, null);
            statement.setTimestamp(11, timestamp);
            statement.setBoolean(12, true);
            statement.setBoolean(13, false);
            statement.setString(14, "New Project Request");
            statement.setString(15, project_type);

            statement.executeUpdate();


            addStage(user.id, nextProjectID, "Waiting Project Verification",
DatabaseConnection.getInstance().getConnection());

            return nextProjectID;

        } catch (SQLException t) {
            System.out.println("SQLException in createProjectRequest()");
        }
        catch (Exception e) {
            System.out.println("Error in ProjectMapper - createProjectRequest()");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return 0;
    }
    public int getNextProjectId() {
        Connection con = null;
        try {
            con = DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {

        }

        PreparedStatement statement = null;
        ResultSet rs = null;
        String SQL = "select MAX(id) from projects";
        int id = 0;

        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
            }

        } catch (Exception e) {
            System.out.println("Error in ProjectMapper - getNextProjectId()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id + 1;
    }

    // Will make following function more generic; should be able to change status depending on parameter to reduce amount
of methods needed
    public boolean changeProjectStatus(int project_id, String new_status, int companyId, int userId,  Connection con) {
        PreparedStatement statement = null;

        String SQL;
        if(companyId == 1) {
            SQL = "UPDATE projects SET status = ?, unread_partner = 1, notification = null where id = ?";
        } else {
            SQL = "UPDATE projects SET status = ?, unread_admin = 1, notification = null where id = ?";
        }
        if (!new_status.equals("Project Approved") && !new_status.equals("Project Finished") &&
!new_status.equals("Cancelled")) {
            try {
                statement = con.prepareStatement(SQL);
                statement.setString(1, new_status);
                statement.setInt(2, project_id);
                statement.executeUpdate();

                updateChangeDate(project_id, companyId);
                addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
                DatabaseFacade facade = new DatabaseFacade();
                if (companyId == 1)
                    facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
                else
                    facade.markUnread(project_id, 1);

                return true;
            } catch (Exception e) {
                System.out.println("Zzzzz");
            } finally {
                if (statement != null) try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                if (con != null) try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
        // IF PROJECT BECOMES APPROVED WE NEED A TRANSACTION TO PROCESS CHANGES IN BUDGET
SIMULTANEOUSLY
        else if (new_status.equals("Project Approved")){
            PreparedStatement budgetStatement = null;

            int currentMonth = Calendar.getInstance().get(Calendar.MONTH);
            int currentYear = Calendar.getInstance().get(Calendar.YEAR);
            int currentQuarter = (currentMonth / 3 ) + 1;

            int project_budget = projectBudgetById(project_id);

            String SQLBudget = "update budget set reserved = reserved + ? where yearnum = ? and quarternum = ?";

            try {
                // TRANSACTION BEGIN
                con.setAutoCommit(false);
                // PROJECT CHANGES
                statement = con.prepareStatement(SQL);
                statement.setString(1, new_status);
                statement.setInt(2, project_id);
                statement.executeUpdate();
                // BUDGET CHANGES
                budgetStatement = con.prepareStatement(SQLBudget);
                budgetStatement.setInt(1, project_budget);
                budgetStatement.setInt(2, currentYear);
                budgetStatement.setInt(3, currentQuarter);
                budgetStatement.executeUpdate();
                // COMMIT
                con.commit();
                // TRANSACTION OVER
                updateChangeDate(project_id, companyId);
                addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
                DatabaseFacade facade = new DatabaseFacade();
                if (companyId == 1)
                    facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
                else
                    facade.markUnread(project_id, 1);

                return true;
            } catch (Exception e) {
                try {
                    con.rollback();
                } catch(Exception y) {
                }
                return false;

            } finally {
                if (statement != null) try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                if (con != null) try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }

        }
        // IF PROJECT IS FINISHED (POE APPROVED) WE NEED A TRANSACTION TO PROCESS CHANGES IN
BUDGET SIMULTANEOUSLY
        else if (new_status.equals("Project Finished")){
            PreparedStatement budgetStatement1 = null;
            PreparedStatement budgetStatement2 = null;

            long timestamp = getProjectFinishedByProjectId(project_id).getTime();
            Calendar cal = Calendar.getInstance();
            cal.setTimeInMillis(timestamp);

            int currentMonth = cal.get(Calendar.MONTH);
            int currentYear = cal.get(Calendar.YEAR);
            int currentQuarter = (currentMonth / 3 ) + 1;

            int project_budget = projectBudgetById(project_id);

            String SQLBudget1 = "update budget set reserved = reserved - ? where yearnum = ? and quarternum = ?";
            String SQLBudget2 = "update budget set reimbursed = reimbursed + ? where yearnum = ? and quarternum = ?";
            try {
                // TRANSACTION BEGIN
                con.setAutoCommit(false);
                // PROJECT CHANGES
                statement = con.prepareStatement(SQL);
                statement.setString(1, new_status);
                statement.setInt(2, project_id);
                statement.executeUpdate();
                // RESERVED CHANGES
                budgetStatement1 = con.prepareStatement(SQLBudget1);
                budgetStatement1.setInt(1, project_budget);
                budgetStatement1.setInt(2, currentYear);
                budgetStatement1.setInt(3, currentQuarter);
                budgetStatement1.executeUpdate();
                // REIMBURSED CHANGES
                budgetStatement2 = con.prepareStatement(SQLBudget2);
                budgetStatement2.setInt(1, project_budget);
                budgetStatement2.setInt(2, currentYear);
                budgetStatement2.setInt(3, currentQuarter);
                budgetStatement2.executeUpdate();
                // COMMIT
                con.commit();
                // TRANSACTION OVER
                updateChangeDate(project_id, companyId);
                addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
                DatabaseFacade facade = new DatabaseFacade();
                if (companyId == 1)
                    facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
                else
                    facade.markUnread(project_id, 1);


                return true;
            } catch (Exception e) {
                try {
                    con.rollback();
                } catch(Exception y) {
                }
                return false;

            } finally {
                if (statement != null) try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                if (con != null) try {
                    con.close();
                } catch (SQLException e) {
```

```java
                e.printStackTrace();
            }
        }

    }
    else if (new_status.equals("Cancelled")){

        String current_status = projectStatusById(project_id);

        if (current_status.equals("Waiting Project Verification") || current_status.equals("Project Rejected") ||
current_status.equals("Project Finished")) {
            try {
                statement = con.prepareStatement(SQL);
                statement.setString(1, new_status);
                statement.setInt(2, project_id);
                statement.executeUpdate();

                updateChangeDate(project_id, companyId);
                addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
                DatabaseFacade facade = new DatabaseFacade();
                if (companyId == 1)
                    facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
                else
                    facade.markUnread(project_id, 1);

                return true;
            } catch (Exception e) {
                System.out.println("Zzzzz");
            } finally {
                if (statement != null) try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                if (con != null) try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        } else {

            PreparedStatement budgetStatement = null;
            String SQLBudget = "update budget set reserved = reserved - ? where yearnum = ? and quarternum = ?";

            long timestamp = getProjectFinishedByProjectId(project_id).getTime();
            Calendar cal = Calendar.getInstance();
            cal.setTimeInMillis(timestamp);

            int currentMonth = cal.get(Calendar.MONTH);
            int currentYear = cal.get(Calendar.YEAR);
            int currentQuarter = (currentMonth / 3) + 1;

            int project_budget = projectBudgetById(project_id);

            try {
                // TRANSACTION BEGIN
                con.setAutoCommit(false);
                // PROJECT CHANGES
                statement = con.prepareStatement(SQL);
                statement.setString(1, new_status);
                statement.setInt(2, project_id);
                statement.executeUpdate();
                // RESERVED CHANGES
                budgetStatement = con.prepareStatement(SQLBudget);
                budgetStatement.setInt(1, project_budget);
                budgetStatement.setInt(2, currentYear);
                budgetStatement.setInt(3, currentQuarter);
                budgetStatement.executeUpdate();
                // COMMIT
                con.commit();
                // TRANSACTION OVER
                updateChangeDate(project_id, companyId);
                addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
                DatabaseFacade facade = new DatabaseFacade();
                if (companyId == 1)
                    facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
                else
                    facade.markUnread(project_id, 1);


                return true;
            } catch (Exception e) {
                try {
                    con.rollback();
                } catch (Exception y) {
                }
                return false;

            } finally {
                if (statement != null) try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                if (con != null) try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }

    }

    return false;

}

public boolean addStage(int user_id, int project_id, String type, Connection con) {
    PreparedStatement statement = null;
    java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
    Timestamp timestamp = new Timestamp(date.getTime());
    int id = getNextStageId();
    String SQL = "insert into stages values(?,?,?,?,?)";

    try {
        statement = con.prepareStatement(SQL);

        statement.setInt(1, id);
        statement.setInt(2, user_id);
        statement.setInt(3, project_id);
        statement.setTimestamp(4, timestamp);
        statement.setString(5, type);

        statement.executeUpdate();

        return true;
    } catch (Exception e) {
        System.out.println("Exception in addStage()");
    } finally {
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
    return false;

}

public ArrayList getStagesByProjectId(int project_id, Connection con) {
    String SQL = "select * from stages where project_id=?";
    ArrayList<Stage> stages = new ArrayList<>();
    PreparedStatement statement = null;
    ResultSet rs = null;

    try {
        statement = con.prepareStatement(SQL);
        statement.setInt(1, project_id);

        rs = statement.executeQuery();
        while(rs.next()) {
            stages.add(new Stage(rs.getInt(1),
                    rs.getInt(2),
                    rs.getInt(3),
                    rs.getTimestamp(4),
                    rs.getString(5)));

        }

    } catch (Exception e) {
        System.out.println("Error in UserMapper");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }


    return stages;
}

public int getNextStageId() {
    PreparedStatement statement = null;
    ResultSet rs = null;
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) {};
    String SQL = "select MAX(id) from stages";
    int id = 0;

    try {
        statement = con.prepareStatement(SQL);
        rs = statement.executeQuery();
        while (rs.next()) {
            id = rs.getInt(1);
        }

    } catch (Exception e) {
        System.out.println("Error in ProjectMapper - getNextStageId()");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return id + 1;
}

public DisplayProject getProjectById(int id, int companyId, Connection con) {
    String SQL = "select * from projects where id=?";
    DisplayProject project = null;
```

```
        PreparedStatement statement = null;
        ResultSet rs = null;

        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, id);

            rs = statement.executeQuery();
            if (rs.next()) {
                project = new DisplayProject(rs.getInt(1),
                        rs.getTimestamp(2),
                        rs.getTimestamp(3),
                        rs.getInt(4),
                        rs.getInt(5),
                        rs.getString(6),
                        rs.getInt(7),
                        rs.getString(8),
                        rs.getTimestamp(9),
                        rs.getTimestamp(10),
                        rs.getTimestamp(11),
                        rs.getBoolean(12),
                        rs.getBoolean(13),
                        rs.getString(14),
                        rs.getString(15)
                );
            } else {
                project.message = "A project with the id " + id + "doesn't exist.";
            }

        } catch (Exception e) {
            System.out.println("Error in UserMapper");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        if(project != null)
            if(companyId != 1)
                if(project.getCompany_id() != companyId) { //unauthorized access
                    project = new DisplayProject();
                    project.message = "You do not have permission to view this project";
                }
        return project;
    }

    public void changeReadStatus(int read, int id, int companyId, Connection con) {
        String SQL = "";
        System.out.println("Read: " + read + ", projid: " + id + ", compId: " + companyId);
        if(companyId == 1)
             SQL = "update projects set unread_admin=? where id=?";
        else
            SQL = "update projects set unread_partner=? where id=?";

        PreparedStatement statement = null;
        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, read);
            statement.setInt(2, id);

            statement.executeUpdate();

        } catch (Exception e) {
            System.out.println("Error in markRead");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

    }

    public ArrayList getProjectsByState(String state, int companyId, Connection con) {
        ArrayList<DisplayProject> displayProjects = new ArrayList<>();

        String SQL;
        if(companyId == 1) { // Request from Dell user
            if(state.equals("waitingForAction"))
                SQL = "select * from projects where status='Waiting Project Verification' or status='Waiting Claim Verification'
order by last_change_partner DESC, start_time DESC";
            else if(state.equals("inExecution"))
                SQL = "select * from projects where status='Project Approved' or status='Project Rejected' or status='Claim
Rejected' order by last_change_partner DESC, start_time DESC";
            else if(state.equals("finished"))
                SQL = "select * from projects where status='Project Finished' or status='Cancelled' order by
last_change_partner DESC, start_time DESC";
            else
                SQL = "select * from projects where status='" + state + "' order by last_change_partner DESC, start_time
DESC";
        } else {
            if(state.equals("waitingForAction"))
                SQL = "select * from projects where status not in ('Project Finished', 'Cancelled') and company_id=? order by
case when last_change_admin is null then 0 else 1 end DESC, last_change_admin DESC, start_time DESC";
            else if(state.equals("finished"))
                SQL = "select * from projects where status in ('Project Finished', 'Cancelled') and company_id=? order by case
when last_change_admin is null then 0 else 1 end DESC, last_change_admin DESC, start_time DESC";
            else
                SQL = "select * from projects where status='" + state + "' and company_id=? order by case when
last_change_admin is null then 0 else 1 end DESC, last_change_admin DESC, start_time DESC";
        }


        PreparedStatement statement = null;
        ResultSet rs = null;

        try {
            statement = con.prepareStatement(SQL);
            if(companyId != 1)
                statement.setInt(1, companyId);

            rs = statement.executeQuery();
            while (rs.next()) {
                displayProjects.add(new DisplayProject(rs.getInt(1),
                        rs.getTimestamp(2),
                        rs.getTimestamp(3),
                        rs.getInt(4),
                        rs.getInt(5),
                        rs.getString(6),
                        rs.getInt(7),
                        rs.getString(8),
                        rs.getTimestamp(9),
                        rs.getTimestamp(10),
                        rs.getTimestamp(11),
                        rs.getBoolean(12),
                        rs.getBoolean(13),
                        rs.getString(14),
                        rs.getString(15)
                ));
            }

        } catch (Exception e) {
            System.out.println("Error in UserMapper");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return displayProjects;
    }

    public ArrayList getProjectsByType(String type, int companyId, Connection con) {
        ArrayList<DisplayProject> displayProjects = new ArrayList<>();

        String SQL;
        if(companyId == 1)
            SQL = "select * from projects where type=? order by last_change_partner DESC, start_time DESC";
        else
            SQL = "select * from projects where type=? and company_id=? order by case when last_change_admin is null
then 0 else 1 end DESC, last_change_admin DESC, start_time DESC";



        PreparedStatement statement = null;
        ResultSet rs = null;

        try {
            statement = con.prepareStatement(SQL);
            statement.setString(1, type);
            if(companyId != 1)
                statement.setInt(2, companyId);

            rs = statement.executeQuery();
            while (rs.next()) {
                displayProjects.add(new DisplayProject(rs.getInt(1),
                        rs.getTimestamp(2),
                        rs.getTimestamp(3),
                        rs.getInt(4),
                        rs.getInt(5),
                        rs.getString(6),
                        rs.getInt(7),
                        rs.getString(8),
                        rs.getTimestamp(9),
                        rs.getTimestamp(10),
                        rs.getTimestamp(11),
                        rs.getBoolean(12),
                        rs.getBoolean(13),
                        rs.getString(14),
                        rs.getString(15)
                ));
            }

        } catch (Exception e) {
            System.out.println("Error in UserMapper");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return displayProjects;
    }

    public ArrayList getProjectsByCompanyName(String companyName, int companyId, Connection con) {
        ArrayList<DisplayProject> displayProjects = new ArrayList<>();

        DatabaseFacade facade = new DatabaseFacade();
        int id = facade.getCompanyIdByName(companyName);

        String SQL = "select * from projects where company_id=? order by last_change_partner DESC, start_time DESC";
```

```java
        PreparedStatement statement = null;
        ResultSet rs = null;

        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, id);

            rs = statement.executeQuery();
            while (rs.next()) {
                displayProjects.add(new DisplayProject(rs.getInt(1),
                        rs.getTimestamp(2),
                        rs.getTimestamp(3),
                        rs.getInt(4),
                        rs.getInt(5),
                        rs.getString(6),
                        rs.getInt(7),
                        rs.getString(8),
                        rs.getTimestamp(9),
                        rs.getTimestamp(10),
                        rs.getTimestamp(11),
                        rs.getBoolean(12),
                        rs.getBoolean(13),
                        rs.getString(14),
                        rs.getString(15)
                ));
            }

        } catch (Exception e) {
            System.out.println("Error in UserMapper");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }


        return displayProjects;
    }

    public int[] getStatusCounts(int companyId, Connection con) {
        PreparedStatement statement = null;
        ResultSet rs = null;
        String SQL = "";
        if(companyId == 1) {
            SQL = "select " +
                    "  sum(case when status='Waiting Project Verification' or status='Waiting Claim Verification' then 1 else 0 end)
WaitingForAction," +
                    "  sum(case when status='Project Approved' or status='Project Rejected' or status='Claim Rejected'  then 1 else
0 end) InExecution," +
                    "  sum(case when status='Project Finished' then 1 else 0 end) Finished" +
                    " from projects";
        } else {
            SQL = "select \n" +
                    "  sum(case when (status not in ('Project Finished', 'Cancelled')) and company_id=" + companyId + " then 1
else 0 end) WaitingForAction,\n" +
                    "  sum(case when status=" and company_id=" + companyId + "  then 1 else 0 end) InExecution,\n" +
                    "  sum(case when status in ('Project Finished', 'Cancelled') and company_id=" + companyId + "   then 1 else 0
end) Finished\n" +
                    " from projects";
        }

        int[] res = new int[3];
        System.out.println(SQL);

        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            if (rs.next()) {
                res[0] = rs.getInt(1);
                res[1] = rs.getInt(2);
                res[2] = rs.getInt(3);
            }
            System.out.println(res[0]);
        } catch (Exception e) {
            System.out.println("Error in ProjectMapper - getStatusCounts()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return res;
    }





    public void updateChangeDate(int project_id, int companyId) {

        Connection con = null;
        try {
            con = DatabaseConnection.getInstance().getConnection();
        } catch (Exception e) {

        }

        if (companyId == 1) {
            java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
            Timestamp timestamp = new Timestamp(date.getTime());
            PreparedStatement statement = null;
            String SQL = "UPDATE projects SET last_change_admin = ? where id = ? ";
            try {
                statement = con.prepareStatement(SQL);
                statement.setTimestamp(1, timestamp);
                statement.setInt(2, project_id);
                statement.executeUpdate();

            } catch (Exception e) {
                System.out.println("Error in updateChangeDate()");
            } finally {
                if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
                if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
            }
        } else {
            java.sql.Date date = new java.sql.Date(System.currentTimeMillis());
            Timestamp timestamp = new Timestamp(date.getTime());
            PreparedStatement statement = null;
            String SQL = "UPDATE projects SET last_change_partner = ? where id = ? ";
            try {
                statement = con.prepareStatement(SQL);
                statement.setTimestamp(1, timestamp);
                statement.setInt(2, project_id);
                statement.executeUpdate();

            } catch (Exception e) {
                System.out.println("Error in updateChangeDate()");
            } finally {
                if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
                if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
            }
```

```java
    }
}

public void updateNotification(int project_id, String notification) {
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) {

    }

    PreparedStatement statement = null;
    String SQL = "UPDATE projects SET notification = ? where id = ? ";
    try {
        statement = con.prepareStatement(SQL);
        statement.setString(1, notification);
        statement.setInt(2, project_id);
        statement.executeUpdate();

    } catch (Exception e) {
        System.out.println("Error in updateNotification()");
    } finally {
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
}

public ArrayList getDistinctStatuses(String query, Connection con) {
    ArrayList<String> results = new ArrayList<>();
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select DISTINCT status\n" +
            "from projects\n" +
            "where lower(status) like lower('%" + query + "%')\n";

    System.out.println(SQL);

    try {
        statement = con.prepareStatement(SQL);
        rs = statement.executeQuery();
        while (rs.next()) {
            results.add(rs.getString(1));
        }
    } catch (Exception e) {
        System.out.println("Error in ProjectMapper - getDistinctStatuses()");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return results;
}

public ArrayList getDistinctTypes(String query, int companyId, Connection con) {
    ArrayList<String> results = new ArrayList<>();
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL;
    if(companyId != 1)
        SQL = "select DISTINCT type\n" +
            "from projects\n" +
            "where company_id=? and lower(type) like lower('%" + query + "%')\n";
    else
        SQL = "select DISTINCT type\n" +
                "from projects\n" +
                "where lower(type) like lower('%" + query + "%')\n";
        System.out.println(SQL);

        try {
            statement = con.prepareStatement(SQL);
            if(companyId != 1)
                statement.setInt(1, companyId);
            rs = statement.executeQuery();
            while (rs.next()) {
                results.add(rs.getString(1));
            }
        } catch (Exception e) {
            System.out.println("Error in ProjectMapper - getDistinctStatuses()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return results;
    }

    public ArrayList search(String query, int companyId, Connection con) {
        ArrayList<Result> results = new ArrayList<>();
        PreparedStatement statement = null;
        ResultSet rs = null;
        String SQL;

        if(companyId == 1)
            SQL = "    select 'Project' OriginatingTable, id, body, utl_match.jaro_winkler_similarity(body, "'" + query + "'") Ayy,
1\n" +
                "    from projects\n" +
                "    where lower(body) like lower('%" + query + "%')\n" +
                "    union all\n" +
                "    select 'Message', project_id, body, utl_match.jaro_winkler_similarity(body, "'" + query + "'") Ayy, 2\n" +
                "    from messages\n" +
                "    where lower(body) like lower('%" + query + "%')\n" +
                "    union all\n" +
                "    select 'User', id, name, utl_match.jaro_winkler_similarity(name, "'" + query + "'") Ayy, 3\n" +
                "    from users\n" +
                "    where lower(name) like lower('%" + query + "%')\n" +
                "    ORDER by 5 ASC, Ayy DESC";
        else
            SQL = "    select 'Project' OriginatingTable, id, body, utl_match.jaro_winkler_similarity(body, "'" + query + "'") Ayy,
1\n" +
                    "    from projects\n" +
                    "    where lower(body) like lower('%" + query + "%') and company_id=" + companyId +"\n" +
                    "    union all\n" +
                    "    select 'Message', project_id, body, utl_match.jaro_winkler_similarity(body, "'" + query + "'") Ayy, 2\n" +
                    "    from messages\n" +
                    "    where lower(body) like lower('%" + query + "%') and author_id IN (SELECT id from users where
company_id=" + companyId + ") \n" +
                    "    union all\n" +
                    "    select 'User', id, name, utl_match.jaro_winkler_similarity(name, "'" + query + "'") Ayy, 3\n" +
                    "    from users\n" +
                    "    where lower(name) like lower('%" + query + "t%') and company_id=" + companyId +"\n" +
                    "    ORDER by 5 ASC, Ayy DESC";


        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                results.add(new Result(
                        rs.getString(1),
                        rs.getInt(2),
                        rs.getString(3),
                    rs.getInt(4)
            ));
        }
    } catch (Exception e) {
        System.out.println("Error in ProjectMapper - search()");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return results;
}


public ArrayList<Project> getProjectsByCompanyId(int company_id, Connection con) {
    String SQL = "select * from projects where company_id = ?";
    PreparedStatement statement = null;
    ResultSet rs = null;
    ArrayList<Project> ProjectCollection = new ArrayList<>();


    try {
        statement = con.prepareStatement(SQL);

        statement.setInt(1, company_id);

        rs = statement.executeQuery();

        while (rs.next()) {
            ProjectCollection.add(new Project(
                        rs.getInt(1),
                        rs.getTimestamp(2),
                        rs.getTimestamp(3),
                        rs.getInt(4),
                        rs.getInt(5),
                        rs.getString(6),
                        rs.getInt(7),
                        rs.getString(8),
                        rs.getTimestamp(9),
                        rs.getTimestamp(10),
                        rs.getTimestamp(11),
                        rs.getBoolean(12),
                        rs.getBoolean(13),
                        rs.getString(14),
                        rs.getString(15)
            ));
        }



    } catch (Exception e) {
        System.out.println("error in projectmapperrerr get by company id method" );
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }


    return ProjectCollection;
}

public ArrayList<Project> getProjectsByUserId(int user_id, Connection con) {
    String SQL = "select * from projects where user_id = ?";
    PreparedStatement statement = null;
    ResultSet rs = null;
```

```java
    ArrayList<Project> ProjectCollection = new ArrayList<>();

    try {
        statement = con.prepareStatement(SQL);
        statement.setInt(1, user_id);

        rs = statement.executeQuery();

        while (rs.next()) {
            ProjectCollection.add(new Project(
                    rs.getInt(1),
                    rs.getTimestamp(2),
                    rs.getTimestamp(3),
                    rs.getInt(4),
                    rs.getInt(5),
                    rs.getString(6),
                    rs.getInt(7),
                    rs.getString(8),
                    rs.getTimestamp(9),
                    rs.getTimestamp(10),
                    rs.getTimestamp(11),
                    rs.getBoolean(12),
                    rs.getBoolean(13),
                    rs.getString(14),
                    rs.getString(15)
            ));
        }

    } catch (Exception e) {
        System.out.println("error in ProjectMapper get by user id method" );
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return ProjectCollection;
}

public int projectBudgetById(int project_id) {
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select budget from projects where id= ? ";
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) {

    }

    try {
        statement = con.prepareStatement(SQL);

        statement.setInt(1, project_id);

        rs = statement.executeQuery();

        while (rs.next()) {
            return rs.getInt(1);
        }

    } catch (Exception e) {
        System.out.println("error irgwgew" );
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return -1;
}

public Timestamp getProjectFinishedByProjectId(int project_id) {
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select stages.time from stages where project_id= ? ";
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) {

    }

    try {
        statement = con.prepareStatement(SQL);

        statement.setInt(1, project_id);

        rs = statement.executeQuery();

        while (rs.next()) {
            return rs.getTimestamp(1);
        }

    } catch (Exception e) {
        System.out.println("error in timestampgetterererer" );
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return null;
}

public String projectStatusById(int project_id) {
    PreparedStatement statement = null;
    ResultSet rs = null;
    String SQL = "select status from projects where id= ? ";
    Connection con = null;
    try {
        con = DatabaseConnection.getInstance().getConnection();
    } catch (Exception e) {

    }

    try {
        statement = con.prepareStatement(SQL);

        statement.setInt(1, project_id);

        rs = statement.executeQuery();

        while (rs.next()) {
            return rs.getString(1);
        }

    } catch (Exception e) {
        System.out.println("error stringGetetERERERERERR status project blebeleb" );
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
```

```
        return null;
    }


}
```

StatisticsGetter

```
package DataLayer;

import java.sql.*;
import java.util.ArrayList;

public class StatisticsGetter {
    public static ArrayList<String[]> getDistinctTypesCounts(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select distinct(type), count(type) as count" +
                " from projects" +
                " where START_TIME between ? and ?" +
                " GROUP BY type";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getDistinctTypesCounts");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
    public static ArrayList<String[]> getAvgCostPerType(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select distinct(type), avg(budget)" +
                " from projects" +
                " where START_TIME between ? and ?" +
                " GROUP BY type";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getAvgCostPerType");
        }finally {
```

```java
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
    return res;
}
public static ArrayList<String[]> getCostPerType(Timestamp start, Timestamp end, Connection con) {

    String SQL = "select distinct(type), sum(budget)" +
            " from projects" +
            " where START_TIME between ? and ?" +
            " GROUP BY type";
    PreparedStatement statement = null;
    ResultSet rs = null;
    ArrayList<String[]> res = new ArrayList<>();


    try {
        statement = con.prepareStatement(SQL);
        statement.setTimestamp(1, start);
        statement.setTimestamp(2, end);
        rs = statement.executeQuery();
        while(rs.next()) {
            res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
        }
    } catch (Exception e) {
        System.out.println("error in StatisticsGetter#getCostPerType");
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
    return res;
}
public static ArrayList<String[]> getBudgetProgression(Timestamp start, Timestamp end, Connection con) {

    String SQL = "select stages.time, projects.budget from projects" +
            " LEFT JOIN STAGES" +
            " on PROJECTS.ID = STAGES.PROJECT_ID" +
            " where stages.type = 'Project Approved'" +
            " and START_TIME between ? and ?" +
            " order by stages.time";
    PreparedStatement statement = null;
    ResultSet rs = null;
    ArrayList<String[]> res = new ArrayList<>();


    try {
        statement = con.prepareStatement(SQL);
        statement.setTimestamp(1, start);
        statement.setTimestamp(2, end);
        rs = statement.executeQuery();
        while(rs.next()) {
            res.add(new String[]{String.valueOf(rs.getTimestamp(1).getTime()), String.valueOf(rs.getInt(2))});
        }
    } catch (Exception e) {
        System.out.println("error in StatisticsGetter#getBudgetProgression");
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }
    return res;
}
public static ArrayList<String[]> getCountOfProjectsByCountry(Timestamp start, Timestamp end, Connection con) {

    String SQL = "select distinct(companies.COUNTRY_CODE), count(PROJECTS.id)" +
                " from companies" +
                " LEFT JOIN PROJECTS" +
                " ON COMPANIES.ID = PROJECTS.COMPANY_ID" +
                " where START_TIME between ? and ?" +
                " GROUP BY COMPANIES.COUNTRY_CODE";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getCountOfProjectsByCountry");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
    public static ArrayList<String[]> getAvgCostOfProjectsByCountry(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select distinct(companies.COUNTRY_CODE), avg(PROJECTS.budget) from companies" +
                " LEFT JOIN PROJECTS" +
                "  ON COMPANIES.ID = PROJECTS.COMPANY_ID" +
                " where START_TIME between ? and ?" +
                " GROUP BY COMPANIES.COUNTRY_CODE";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getAvgCostOfProjectsByCountry");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
    public static ArrayList<String[]> getCompaniesByLargestApprovedBudget(Timestamp start, Timestamp end, Connection
con) {

        String SQL = "select C.name, pbudget" +
                "  from (" +
                "        select distinct(c.id) cid, sum(p.budget) pbudget from companies c" +
                "          LEFT JOIN PROJECTS p" +
                "            ON c.ID = p.COMPANY_ID" +
                "          where p.budget is not null and p.status not in ('Waiting Project Verification', 'Project Rejected') and
p.START_TIME between ? and ?" +
                "        GROUP BY c.id) lists" +
                " INNER JOIN companies C on lists.cid = C.id" +
                " ORDER BY pbudget DESC";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getInt(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getCompaniesByLargestApprovedBudget");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
    public static ArrayList<String[]> getTypesWithHighestSuccessRate(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select distinct(projects.type), cast(count(c.TYPE) as FLOAT)/cast(count(PROJECTS.type) as float) as
Percent from projects" +
                " left join (select * from projects where status in ('Project Finished','Reimbursed')) C" +
                "  ON PROJECTS.id = c.id" +
                " where projects.START_TIME between ? and ?" +
                " group by projects.type" +
                " order by Percent DESC";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{rs.getString(1), String.valueOf(rs.getDouble(2))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getTypesWithHighestSuccessRate");
            System.out.println(e.getMessage());
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
public static ArrayList<String[]> getCountOfMessages(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select count(id)" +
                " from MESSAGES" +
                " where CREATION_TIME between ? and ?";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{String.valueOf(rs.getInt(1))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getCountOfMessages");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
public static ArrayList<String[]> getAvgMessagesPerProject(Timestamp start, Timestamp end, Connection con) {

        String SQL = "Select tot2/tot1" +
                " from" +
                "  (Select Count(id) as tot1 FROM projects where START_TIME between ? and ?) h," +
                " (Select Count(id) as tot2 FROM MESSAGES where CREATION_TIME between ? and ?) s";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            statement.setTimestamp(3, start);
            statement.setTimestamp(4, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{String.valueOf(rs.getInt(1))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getAvgMessagesPerProject");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
public static ArrayList<String[]> getCountOfFinishedProjects(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select count(id)" +
                " from projects" +
                " where STATUS='Project Finished'" +
                " and START_TIME between ? and ?";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            statement.setTimestamp(1, start);
            statement.setTimestamp(2, end);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{String.valueOf(rs.getInt(1))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getCountOfFinishedProjects");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }
public static ArrayList<String[]> getMoneyReimbursed(Timestamp start, Timestamp end, Connection con) {

        String SQL = "select sum(REIMBURSED) from BUDGET";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<String[]> res = new ArrayList<>();

        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while(rs.next()) {
                res.add(new String[]{String.valueOf(rs.getInt(1))});
            }
        } catch (Exception e) {
            System.out.println("error in StatisticsGetter#getMoneyReimbursed");
            System.out.println(e.getMessage());
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }
        return res;
    }

}
```

UserMapper

```
package DataLayer;

import java.sql.*;
import java.util.ArrayList;

import Domain.*;

public class UserMapper {

    public User getUserById(int user_id, Connection con) {

        User user = null;
        String SQL = "select * from users where id=?";

        PreparedStatement statement = null;
        ResultSet rs = null;
        try {
            statement = con.prepareStatement(SQL);

            statement.setInt(1, user_id);
            rs = statement.executeQuery();
            if (rs.next())
            {
                user = new User(rs.getInt(1),
                        rs.getString(2),
                        rs.getString(3),
                        rs.getString(4),
                        rs.getInt(5),
                        rs.getBoolean(6)
                        );
```

```java
        }

    } catch (Exception e) {
        System.out.println("Error in UserMapper");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return user;
}

public User getUserByEmail(String email, Connection con) {
    User user = null;
    String SQL = "select * from users where lower(email)=?  and deleted = 0";

    PreparedStatement statement = null;
    System.out.println(SQL);

    ResultSet rs = null;
    try {
        statement = con.prepareStatement(SQL);
        statement.setString(1, email.toLowerCase());
        rs = statement.executeQuery();
        if (rs.next())
        {
            user = new User(rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4),
                    rs.getInt(5),
                    rs.getBoolean(6));
        }

    } catch (Exception e) {
        System.out.println("Error in UserMapper");
    } finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }


    return user;
}

public int createUser(String name, String user_email, int company_id, Connection con) {
    String SQL = "insert into users values (?, ?, ?, ? ,?, ?)";

    PreparedStatement statement = null;

    try {
        statement = con.prepareStatement(SQL);
        int nextUserId = getNextUserId(con);

        statement.setInt(1, nextUserId);
        statement.setString(2, name);
        statement.setString(3, user_email);
        statement.setString(4, null);
        statement.setInt(5, company_id);
        statement.setInt(6, 0);

        statement.executeUpdate();

        return nextUserId;
    } catch (Exception e) {
            System.out.println("error in user creation");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return -1;
    }

    public int getNextUserId(Connection con) {
        PreparedStatement statement = null;
        String SQL = "select MAX(id) from users";
        int id = 0;

        ResultSet rs = null;
        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();
            while (rs.next()) {
                id = rs.getInt(1);
            }

        } catch (Exception e) {
            System.out.println("Error in UserMapper - getNextUserId()");
        } finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return id + 1;
    }


    public ArrayList<User> getUsersByCompanyId(int company_id, Connection con) {

            String SQL = "select * from users where company_id = ?  and deleted = 0";
            PreparedStatement statement = null;
            ResultSet rs = null;
            ArrayList<User> UserCollection = new ArrayList<>();


            try {
                statement = con.prepareStatement(SQL);

                statement.setInt(1, company_id);

                rs = statement.executeQuery();

                while (rs.next()) {
                    UserCollection.add(new User(rs.getInt(1),
                            rs.getString(2),
                            rs.getString(3),
                            rs.getString(4),
                            rs.getInt(5),
                            rs.getBoolean(6)
                    ));
                }

            } catch (Exception e) {
                System.out.println("error in budgetmapperrerr");
            }finally {
                if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
                if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
                if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
            }
        return UserCollection;
    }

    public ArrayList<User> getUsers(Connection con) {

        String SQL = "select * from users where deleted = 0";
        PreparedStatement statement = null;
        ResultSet rs = null;
        ArrayList<User> UserCollection = new ArrayList<>();


        try {
            statement = con.prepareStatement(SQL);
            rs = statement.executeQuery();

            while (rs.next()) {
                UserCollection.add(new User(rs.getInt(1),
                        rs.getString(2),
                        rs.getString(3),
                        rs.getString(4),
                        rs.getInt(5),
                        rs.getBoolean(6)
                ));
            }

        } catch (Exception e) {
            System.out.println("error in UserMapper - getUsers");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return UserCollection;
    }

    public boolean createPassword(int id, String password, Connection con) {
        String SQL = "update users set password=? where id=?";

        PreparedStatement statement = null;

        System.out.println(id +", " + password);

        try {
            statement = con.prepareStatement(SQL);
            statement.setString(1, password);
            statement.setInt(2, id);

            statement.executeUpdate();

            return true;
        } catch (Exception e) {
            System.out.println("error in createPassword, in mapper");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return false;
    }

    public ArrayList getUserInfoInvolvedInProjectById(int project_id, int user_id, Connection con) {
        ArrayList<String[]> emails = new ArrayList<>();
        PreparedStatement statement = null;
        ResultSet rs = null;

        String SQL = "select email, name from users \n" +
```

```java
            "where id <> ? and deleted = 0 and \n" +
            "  id in (select user_id from stages where project_id=?) or\n" +
            "  id in (select author_id from messages where project_id=?)";

        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, user_id);
            statement.setInt(2, project_id);
            statement.setInt(3, project_id);
            rs = statement.executeQuery();

            while (rs.next()) {
                emails.add(new String[]{rs.getString(1), rs.getString(2)});
            }



        } catch (Exception e) {
            System.out.println("error in UserMapper - getUserInfoInvolvedInProjectById()");
        }finally {
            if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return emails;

    }

    public boolean markUserDeleted(int user_id, Connection con) {
        PreparedStatement statement = null;

        String SQL = "update users set deleted = '1' where id = ?";

        try {
            statement = con.prepareStatement(SQL);
            statement.setInt(1, user_id);

            statement.executeUpdate();

            return true;
        } catch (Exception e) {
            System.out.println("error in markuserdeleted, in mapper");
        } finally {
            if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
            if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
        }

        return false;

    }
}
```

Budget

```java
package Domain;

public class Budget {
    int initial_budget;
    int year;
    int quarter;
    int reserved;
    int reimbursed;

    public Budget(int initial_budget, int year, int quarter, int reimbursed, int reserved) {
        this.initial_budget = initial_budget;
        this.year = year;
        this.quarter = quarter;
        this.reimbursed = reimbursed;
        this.reserved = reserved;
    }

    public int getInitial_budget() {
        return initial_budget;
    }

    public int getYear() {
        return year;
    }

    public int getQuarter() {
        return quarter;
    }

    public int getReimbursed() {
        return reimbursed;
    }

    public int getReserved() {
        return reserved;
    }

    public int getLeftAvailable() {
        return initial_budget - reserved - reimbursed;
    }
}
```

Company

```
package Domain;

public class Company {

    int id;
    String name;
    String img_filename;
    String country_code;
    public Company(int id, String name, String img_filename, String country_code) {
        this.id = id;
        this.name = name;
        this.img_filename = img_filename;
        this.country_code = country_code;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getImg_filename() {
        return img_filename;
    }

    public String country_code() {
        return country_code;
    }
}
```

Controller

```
package Domain;

import DataLayer.DatabaseFacade;


import java.net.URL;
import java.net.URLConnection;
import java.nio.channels.Channels;
import java.nio.channels.ReadableByteChannel;
import java.sql.Time;
import java.sql.Timestamp;

import java.util.*;

import javax.servlet.http.Part;
import java.io.*;

public class Controller {

    DatabaseFacade facade;
    IdGenerator gen;

    public Controller() {
        this.facade = new DatabaseFacade();
        this.gen = new IdGenerator();
    }
    // Writers
    public int createProjectRequest(int budget, String project_body, User user, String project_type, Timestamp
execution_date) {
        return facade.createProjectRequest(budget, project_body, user, project_type, execution_date);
    }
    public int createCompany(String company_name, String country_code, Part logo, String logo_url) {
        int company_id = facade.createCompany(company_name, country_code);
        if(company_id != -1) { //if success
            String filename = "";
            if(logo != null) {
                FileHandling handler = new FileHandling();

                try {
                    handler.putLogo(logo, company_id);
                } catch(Exception e) {}

                filename = handler.getFileName();
                facade.updateCompanyLogo(filename, company_id);
            } else if(logo_url != null) {
                System.out.println(new File(System.getenv("POE_FOLDER") + File.separator + "companies" + File.separator +
company_id + File.separator + "logo." + logo_url.substring(logo_url.lastIndexOf(".") + 1, logo_url.length())).getPath());
                try {
                    URL website = new URL(logo_url);
                    ReadableByteChannel rbc = Channels.newChannel(website.openStream());
                    File outputFolder = new File(System.getenv("POE_FOLDER") + File.separator + "companies" + File.separator
+ company_id );
                    outputFolder.mkdirs();
                    String ext = logo_url.substring(logo_url.lastIndexOf(".") + 1);
                    if(ext.equals("png") || ext.equals("jpg") || ext.equals("jpeg") || ext.equals("gif") || ext.equals("bmp"))
                        filename = company_name + "-logo." + ext;
                    else {
                        URLConnection conn = website.openConnection();
                        String type = conn.getContentType();
                        System.out.println("type: " + type);
                        if(type.contains("image")) {
                            ext = type.substring(6);
                        } else
                            ext = null;
                    }
```

```java
                    if(ext != null) {
                        filename = company_name + "-logo." + ext;
                        System.out.println(website.toString());
                        System.out.println(filename);
                        FileOutputStream fos = new FileOutputStream(
                                new File(outputFolder.getAbsolutePath() + File.separator + filename));
                        fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
                        fos.close();
                    }
                } catch (IOException e) {
                    System.out.println(e.getMessage());
                    System.out.println("Error in saving logo from url");
                }
                facade.updateCompanyLogo(filename, company_id);
            }


            return company_id;
        } else {
            return -1;
        }
    }

    public boolean changeProjectStatus(int project_id, String current_status, String answer, int companyId, int userId) {
        String new_status = "";
        if(answer.equals("approved")) {
            if(current_status.equals("Waiting Project Verification"))
                new_status = "Project Approved";
            else if(current_status.equals("Project Approved"))
                new_status = "Waiting Claim Verification";
            else if(current_status.equals("Waiting Claim Verification"))
                new_status = "Project Finished";
            else if(current_status.equals("Claim Rejected"))
                new_status = "Waiting Claim Verification";

        } else if(answer.equals("denied")) {
            if(current_status.equals("Waiting Project Verification"))
                new_status = "Project Rejected";
            else if(current_status.equals("Waiting Claim Verification"))
                new_status = "Claim Rejected";

        } else if(answer.equals("cancelled")) {
            new_status = "Cancelled";
        }

        if(facade.changeProjectStatus(project_id, new_status, companyId, userId)) {
            ArrayList<String[]> users = facade.getUserInfoInvolvedInProjectById(project_id, userId);

            for (String[] user : users) {
                if(user[0].matches("^(\\w)+@(\\w)+\\.(\\w){2,}$"))
                    sendEmail(user[0], "New Status for project #" + project_id, Notifications.createNotificationHTML(user[1],
project_id, new_status));
            }
            return true;
        } else
            return false;

    }
    //User related
    public int createUser(String name, String user_email, int company_id) {
        int id = facade.createUser(name, user_email, company_id);

        if(id != -1)
                createPasswordResetNonce(id, user_email);

        return id;
    }
```

```
  //Readers
   //User related
   public User getUserById(int user_id) {
       User user = facade.getUserById(user_id);
       user.company = getCompanyById(user.getCompany_id());
       return user;
   }
   public ArrayList<User> getUsersByCompanyId(int company_id) {
       return facade.getUsersByCompanyId(company_id);
   }
   public ArrayList<User> getUsers() {return facade.getUsers();}
   public User getUserByEmail(String email) {return facade.getUserByEmail(email);}

   public boolean markUserDeleted(int user_id) {
       return facade.markUserDeleted(user_id);
   }

   //Project related
   public DisplayProject getProjectById(int id, int companyId) {
       DisplayProject dp = facade.getProjectById(id, companyId);
       if(dp == null)
           return null;
       if((companyId == 1 && dp.isUnread_admin()) || (companyId != 1 && dp.isUnread_partner()))
           facade.markRead(id, companyId);
       return  dp; }
   public ArrayList getStagesByProjectId(int project_id) {
       return proccessStages(facade.getStagesByProjectId(project_id)); }
   public ArrayList getMessagesByProjectId(int projId) {
       return processMessages(facade.getMessagesByProjectId(projId)); }

   public String postMessage(int userId, int projId, String body, int companyId) { return
processMessage(facade.postMessage(userId, projId, body, companyId)).toHTML();}
   public ArrayList getProjectsByState(String state, int companyId) {
       return  facade.getProjectsByState(state, companyId); }
   public ArrayList getProjectsByType(String type, int companyId) {
       return  facade.getProjectsByType(type, companyId); }
   public ArrayList getProjectsByCompanyName(String companyName, int companyId) {
       return facade.getProjectsByCompanyName(companyName, companyId); }


   // Statistics
   public ArrayList<String> getStatistics(String quarter) {
       int y, q, sM, eM;
       Timestamp start, end;
       Calendar cal = new GregorianCalendar();
       if(quarter != null) {
           y=Integer.parseInt(quarter.substring(0,4));
           q=Integer.parseInt(quarter.substring(5));
           start = quarterToTimestamp(y, q, false);
           end = quarterToTimestamp(y, q, true);
       } else {
           //use current quarter
           int m = Calendar.getInstance().get(Calendar.MONTH);
           y = Calendar.getInstance().get(Calendar.YEAR);
           int currentQuarter = (m / 3 ) + 1;
           start = quarterToTimestamp(y, currentQuarter, false);
           end = quarterToTimestamp(y, currentQuarter, true);
       }
       ArrayList<String> stats = new ArrayList<>();

       stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getBudgetProgression(start, end), "Budget
Progression,line"));
       stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getCountOfProjectsByCountry(start, end),
"Projects by country,geomap"));
       stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getCompaniesByLargestApprovedBudget(start,
end), "Companies with largest approved budget,donut"));
```

```
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getAvgCostPerType(start, end), "Average cost per
type,donut"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getAvgCostOfProjectsByCountry(start, end),
"Average cost of projects by country,donut"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getCostPerType(start, end), "Total cost of each
type,donut"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getTypesWithHighestSuccessRate(start, end),
"Types with highest success rate,donut"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getDistinctTypesCounts(start, end), "Count of
projects for each type,donut"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getAvgMessagesPerProject(start, end), "Average
messages per project,number"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getCountOfFinishedProjects(start, end), "Number
of finished projects,number"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getCountOfMessages(start, end), "Messages
sent,number"));
        stats.add(JSONTranslator.stringArrayArrayListWithOptions(facade.getMoneyReimbursed(start, end), "Money
reimbursed,number"));


        return stats;
    }

    //Search
    public ArrayList search(String q, int companyId) {
        ArrayList<Result> res = facade.search(q, companyId);
        ArrayList<ResultsContainer> container = new ArrayList<>();
        String t;
        boolean skip = false;
        for (Result r : res) {
            skip = false;
            t = r.getType();
            for (int i = 0; i < container.size(); i++) {
                if(container.get(i).getType().equals(t)) {
                    container.get(i).getContainer().add(r);
                    skip = true;
                    break;
                }
            }
            if(skip)
                continue;
            container.add(new ResultsContainer(t));
            container.get(container.size() - 1).getContainer().add(r);
        }

        return container;
    }

    //public boolean changeProjectStatus(String project_id, String new_status, String usertype) { return
facade.verifyProjectRequest(project_id, new_status, usertype); }
    public int[] getStatusCounts(int companyId) { return facade.getStatusCounts(companyId); }


    public ArrayList<Project> getProjectsByCompanyId(int company_id) {
        return facade.getProjectsByCompanyId(company_id);
    }
    public ArrayList<Project> getProjectsByUserId(int company_id) {
        return facade.getProjectsByUserId(company_id);
    }

    public Company getCompanyById(int id) { return facade.getCompanyById(id); }

    //User Login / Registration
    public User login(String email, String password) {
        User user = facade.getUserByEmail(email);
        if(user != null) {
            if (Login.testPassword(password, user.password)) {
```

```
            user.company = facade.getCompanyById(user.getCompany_id()); // Assign company to user
            return user;
        }
        else
            return null;
    }
    return null;
}
// Companies
public ArrayList<Company> getCompanies() {
    return facade.getCompanies();
}
public String getCompanyNames(String query, int companyId) {
    return JSONTranslator.stringArrayList(facade.getCompanyNames(query, companyId));
}

public ArrayList proccessStages(ArrayList stages) {
    HashMap userMap = new HashMap();
    HashMap companyMap = new HashMap();
    User user = null;
    for (Stage s : (ArrayList<Stage>) stages) {

        if(userMap.containsKey(s.user_id))
            s.user = (User) userMap.get(s.user_id);
        else {
            user = facade.getUserById(s.user_id);
            s.user = user;
            userMap.put(s.user_id, user);
        }
        if(companyMap.containsKey(s.user.getCompany_id()))
            s.user.company = (Company) companyMap.get(s.user.getCompany_id());
        else {
            s.user.company = facade.getCompanyById(s.user.getCompany_id());
            companyMap.put(s.user.getCompany_id(), s.user.company);
        }
    }

    Collections.sort(stages, Stage.TIME);

    return stages;

}



public ArrayList processMessages(ArrayList messages) {
    HashMap companyMap = new HashMap();
    HashMap userMap = new HashMap();
    User user = null;
    Company company = null;
    for (Message m : (ArrayList<Message>) messages) {
        if(userMap.containsKey(m.author_id))
            m.user = (User) userMap.get(m.author_id);
        else {
            user = facade.getUserById(m.author_id);
            m.user = user;
            userMap.put(m.author_id, user);
            company = facade.getCompanyById(user.getCompany_id());
            companyMap.put(user.getCompany_id(), company);
        }
        if(m.company == null) {
            if(companyMap.containsKey(m.user.getCompany_id()))
                m.company = (Company) companyMap.get(m.user.getCompany_id());
            else {
                company = facade.getCompanyById(m.user.getCompany_id());
                m.company = company;
                companyMap.put(company.id, company);
            }

        }
    }

    Collections.sort(messages, Message.TIME);

    return messages;

}

public Message processMessage(Message m) {
    m.user = facade.getUserById(m.getAuthor_id());
    m.company = facade.getCompanyById(m.user.getCompany_id());
    return m;
}
// POE
public boolean addPoeFile(int project_id, Part file, int user_id, int stage) throws IOException {
    FileHandling handler = new FileHandling();

    handler.putFile(file, project_id);
    String filename = handler.getFileName();
    String filetype = handler.getFileType();

    System.out.println(project_id);
    System.out.println(filename);
    System.out.println(filetype);
    System.out.println(user_id);

    return facade.addPoeFile(project_id, filename, user_id, filetype, stage);

}

public ArrayList<Poe> getPoe(int project_id) {

    return facade.getPoe(project_id);
}

public boolean deleteFile(String filename, int project_id, int fileId, boolean deleteFile) throws IOException {
    FileHandling handler = new FileHandling();
    if(deleteFile) {
        if (!handler.deleteFile(filename, project_id))
            return false;
        return facade.deletePoe(fileId);
    } else {
        return facade.markDeletePoe(fileId);
    }
}

public void sendEmail(String recipient, String subject, String body) {
    new Notifications().sendEmail(recipient, subject, body);
}

public String getDistinctStatuses(String query) {
    return JSONTranslator.stringArrayList(facade.getDistinctStatuses(query));
}
public String getDistinctTypes(String query, int companyId) {
    return JSONTranslator.stringArrayList(facade.getDistinctTypes(query, companyId));
}

// BUDGETSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
public boolean addBudget(int year, int quarter, int budget) {
    return facade.addBudget(year, quarter, budget);
}

public boolean modifyBudget(int new_budget, int year, int quarter) {
    return facade.modifyBudget(new_budget, year, quarter);
    }

    public Budget getActiveBudget(int year, int quarter) {
        return facade.getActiveBudget(year, quarter);
    }

    public ArrayList<Budget> getAllBudgets() {
        return facade.getAllBudgets();
    }

    public int getAvailableFunds(int year, int quarter) {
        return facade.getAvailableFunds(year, quarter);
    }


    //Nonce / Email
    public void createPasswordResetNonce(int id, String email){
        Nonce nonce = new Nonce(-1, gen.nextNonce(), id, null, "PasswordReset");
        int nonceId = facade.addNonce(nonce);
        User user = facade.getUserById(id);
        if(email != null && nonceId != -1)
            sendEmail(email, "Welcomes!", Notifications.createWelcomeHTML(user.getName() , "http://localhost:8080/reset-
password?n=" + nonce.getNonce()));
    }

    public int getUserIdByNonce(String nonce) {
        return facade.getUserIdByNonce(nonce);
    }

    public boolean createPassword(int id, String password, String nonce) {
        if(facade.createPassword(id, Login.createPassword(password))) {
            facade.deleteNonce(nonce);
            return true;
        } else {
            return false;
        }
    }

    public Timestamp quarterToTimestamp(int y, int q, boolean endOfQuarter) {
        Calendar cal = new GregorianCalendar();
        int m;
        if(endOfQuarter)
            m = q*(12/4);
        else
            m = (q-1)*(12/4) + 1;
        cal.set(Calendar.YEAR, y);
        cal.set(Calendar.MONTH, m);
        if(endOfQuarter)
            cal.set(Calendar.DAY_OF_MONTH, cal.getActualMaximum(Calendar.DAY_OF_MONTH));
        else
            cal.set(Calendar.DAY_OF_MONTH, 1);
        return new Timestamp(cal.getTimeInMillis());
    }

}
```

DisplayProject

```
package Domain;

import java.sql.Timestamp;
import java.util.ArrayList;

public class DisplayProject extends Project{
        public long f_start_time;
    public long f_end_time;
```

```java
    public long f_execution_date;
    public long f_last_change_admin;
    public long f_last_change_partner;

    public String companyName;
    public String companyLogoUrl;

    public String message;

    public DisplayProject() {

    }

    public DisplayProject(int id, Timestamp start_time, Timestamp end_time, int company_id, int owner_id, String status,
double budget, String body, Timestamp execution_date, Timestamp last_change_admin, Timestamp last_change_partner,
boolean unread_admin, boolean unread_partner, String notification, String type) {
        super(id, start_time, end_time, company_id, owner_id, status, budget, body, execution_date, last_change_admin,
last_change_partner, unread_admin, unread_partner, notification, type);

        this.body = "<p>" + body.replaceAll("\\n","</p><p>")  + "</p>";

        if(start_time != null) f_start_time = start_time.getTime();
        if(end_time != null) f_end_time = end_time.getTime();
        if(execution_date != null) f_execution_date = execution_date.getTime();
        if(last_change_admin != null) f_last_change_admin = last_change_admin.getTime();
        if(last_change_partner != null) f_last_change_admin = last_change_partner.getTime();
        Company cp = new Controller().getCompanyById(company_id);

        companyName = cp.name;
        companyLogoUrl = cp.img_filename;
    }

    public long getF_start_time() {
        return f_start_time;
    }

    public long getF_end_time() {
        return f_end_time;
    }

    public long getF_execution_date() {
        return f_execution_date;
    }

    public long getF_last_change_admin() {
        return f_last_change_admin;
    }

    public long getF_last_change_partner() {
        return f_last_change_partner;
    }

    public String getCompanyName() {
        return companyName;
    }

    public String getCompanyLogoUrl() {
        return companyLogoUrl;
    }

    public String getMessage() { return message; }
}
```

FileHandling

```
package Domain;

import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.Part;
import java.io.*;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;


@MultipartConfig
public class FileHandling {

    String filename;
    String filetype;

    public void putFile(Part file, int project_id) throws IOException {

       // String path = System.getProperty("user.dir");
       // String projectRoot = path.substring(0, path.lastIndexOf("\\")) + "\\Poe\\";
       // String PoeRootByProjectId = projectRoot + project_id;

        String path = System.getenv("POE_FOLDER");

        String newPath = path + File.separator + project_id;

        File dir = new File(path + File.separator + project_id);

        dir.mkdir();

        filename = getFileName(file);
        filename = filename.replaceAll(" ", "_");
        filetype = filename.substring(filename.lastIndexOf(".") + 1, filename.length());

        OutputStream out = null;
        InputStream filecontent = null;

        try {
            out = new FileOutputStream(new File(newPath + File.separator
                    + filename));
            filecontent = file.getInputStream();

            int read = 0;
            final byte[] bytes = new byte[1024];

            while ((read = filecontent.read(bytes)) != -1) {
                out.write(bytes, 0, read);
            }

        } catch (FileNotFoundException fne) {

        } finally {
            if (out != null) {
                out.close();
            }
            if (filecontent != null) {
                filecontent.close();
            }

        }
    }
```

```java
public void putLogo(Part file, int company_id) throws IOException {
    String path = System.getenv("POE_FOLDER");

    String newPath = path + File.separator + "companies" + File.separator + company_id;

    File dir = new File(newPath);

    dir.mkdir();

    filename = getFileName(file);
    filetype = filename.substring(filename.lastIndexOf(".") + 1, filename.length());

    OutputStream out = null;
    InputStream filecontent = null;

    try {
        out = new FileOutputStream(new File(newPath + File.separator
                + filename));
        filecontent = file.getInputStream();

        int read = 0;
        final byte[] bytes = new byte[1024];

        while ((read = filecontent.read(bytes)) != -1) {
            out.write(bytes, 0, read);
        }

    } catch (FileNotFoundException fne) {

    } finally {
        if (out != null) {
            out.close();
        }
        if (filecontent != null) {
            filecontent.close();
        }

    }
}

public boolean deleteFile(String filename, int project_id) throws IOException {
   try {
       String stringpath = System.getenv("POE_FOLDER") + File.separator + project_id + File.separator + filename;
       Path path = Paths.get(stringpath);
       Files.delete(path);
   } catch (Exception e) {
       System.out.println("didnt delete file");
       return false;
   }
    return true;
}


public String getFileName(Part file) {

    for (String content : file.getHeader("content-disposition").split(";")) {
        if (content.trim().startsWith("filename")) {
            return content.substring(
                    content.indexOf('=') + 1).trim().replace("\"", "");
        }
    }
    return null;

}

public String getFileName() {
        return filename;
    }
    public String getFileType() {
        return filetype;
    }

}
```

IdGenerator

```
package Domain;

import java.math.BigInteger;
import java.security.SecureRandom;

    public final class IdGenerator {
        private SecureRandom random = new SecureRandom();

        public String nextNonce() {
            return new BigInteger(130, random).toString(32);
        }
    }
```

JSONTranslator

```
package Domain;

import java.lang.reflect.Array;
import java.util.ArrayList;

public class JSONTranslator {

    public static String stringArrayList(ArrayList list) {
        String res = "[";

        for (String item : (ArrayList<String>) list) {
            res += "\"" + item + "\",";
        }
        res = res.substring(0, res.length()-1);

        return res + "]";
    }

    public static String stringArrayArrayList(ArrayList<String[]> list) {
        String res = "[";
        for(String[] itemSet : list) {
            res += "[";
            for(String item : itemSet) {
                if(item.matches("^[\\d\\.]+$"))
                    res += item + ",";
                else
                    res += "\"" + item + "\",";
            }
            res = res.substring(0, res.length()-1);
            res += "],";
        }
        res = res.substring(0, res.length()-1);

        return res + "]";
    }
    public static String stringArrayArrayListWithOptions(ArrayList<String[]> list, String o) {
        String res = stringArrayArrayList(list);
        String options = "";
        for(int i = 0; i < o.split(",").length; i++)
            options += "\"" + o.split(",")[i] + "\",";
```

```java
        options = options.substring(0, options.length() - 1);
        res = res.substring(0,1) + "[" + options + "]," + res.substring(1);
        return res;
    }

}
```

Login

```java
package Domain;

import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.Random;

public class Login {

    public static String createPassword(String pw) {
        // Generate Salt
        Random r = new SecureRandom();
        byte[] s = new byte[6];
        r.nextBytes(s);
        String salt = byteArrayToString(s);

        String password = salt + "$" + stringToHash(salt+pw);

        return password;
    }

    public static boolean testPassword(String pw, String saltedpw) {
        String salt = saltedpw.substring(0, 12);
        return saltedpw.equals(salt + "$" + stringToHash(salt + pw));
    }

    public static String byteArrayToString(byte[] a) {
        char[] HEX_CHARS = "0123456789ABCDEF".toCharArray();

        StringBuilder sb = new StringBuilder(a.length * 2);
        for (byte b : a) {
            sb.append(HEX_CHARS[(b & 0xF0) >> 4]);
            sb.append(HEX_CHARS[b & 0x0F]);
        }
        return sb.toString();
    }

    public static String stringToHash(String s) {
        byte[] hash = null;
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            hash = digest.digest(s.getBytes("UTF-8"));
        } catch (Exception E) {};

        return byteArrayToString(hash);
    }
}
```

Message

```java
package Domain;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Comparator;
```

```
public class Message {

    public int id;
    public int author_id;
    public int project_id;
    public String body;
    public Timestamp creation_date;

    public Long creation_date_millis;

    public User user;
    public Company company;

    public Message(int id, int author_id, int project_id, String body, Timestamp creation_date) {
        this.id = id;
        this.author_id = author_id;
        this.project_id = project_id;
        this.body = "<p>" + body.replaceAll("\\n","</p><p>")  + "</p>";
        this.creation_date = creation_date;

        if(creation_date != null) creation_date_millis = creation_date.getTime();
    }

    public String toHTML() {
        String html = "      <div class=\"item message pull-right\">\n" +
                "                <span class=\"user-data\">" + user.getName() + " - " + company.getName() + "</span>\n" +
                "                <span class=\"date isDate\">" + creation_date_millis + "</span>\n" +
                "                <div class=\"inner-bubble\">" +
                "                    <p>" + body + "</p>\n" +
                "                </div>" +
                "            </div>";


        return html;
    }

    public static final Comparator<Message> TIME = (Message o1, Message o2) ->
o1.creation_date_millis.compareTo(o2.creation_date_millis);

    public int getId() {
        return id;
    }

    public int getAuthor_id() {
        return author_id;
    }

    public int getProject_id() {
        return project_id;
    }

    public String getBody() {
        return body;
    }

    public Timestamp getCreation_date() {
        return creation_date;
    }

    public Long getCreation_date_millis() {
        return creation_date_millis;
    }

    public User getUser() {
        return user;
    }
    public Company getCompany() {
        return company;
    }
}
```

Nonce

```
package Domain;

import java.sql.Timestamp;

public class Nonce {

    public int id;
    public String nonce;
    public int associate_id;
    public Timestamp timestamp;
    public String type;

    public Nonce() {

    }

    public Nonce(int id, String nonce, int associate_id, Timestamp timestamp, String type) {
        this.id = id;
        this.nonce = nonce;
        this.associate_id = associate_id;
        this.timestamp = timestamp;
        this.type = type;
    }

    public int getId() {
        return id;
    }

    public String getNonce() {
        return nonce;
    }

    public int getAssociate_id() {
        return associate_id;
    }

    public Timestamp getTimestamp() {
        return timestamp;
    }

    public String getType() {
        return type;
    }
}
```

Notifications

```
package Domain;

    import java.util.Properties;

    import javax.mail.Message;
    import javax.mail.MessagingException;
    import javax.mail.Session;
    import javax.mail.Transport;
    import javax.mail.internet.AddressException;
    import javax.mail.internet.InternetAddress;
```

```java
import javax.mail.internet.MimeMessage;
import java.io.File;
import java.io.FileWriter;
import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;


public class Notifications {
    public void sendEmail(String recipient, String subject, String html){

        try{
            final String fromEmail = "automessagejava@gmail.com"; //requires valid gmail id
            final String password = "sappword123"; // correct password for gmail id
            final String toEmail = recipient; // can be any email id

            System.out.println("TLSEmail Start");
            Properties props = new Properties();
            props.put("mail.smtp.host", "smtp.gmail.com"); //SMTP Host
            props.put("mail.smtp.port", "587"); //TLS Port
            props.put("mail.smtp.auth", "true"); //enable authentication
            props.put("mail.smtp.starttls.enable", "true"); //enable STARTTLS
            props.put("mail.smtp.ssl.trust", "smtp.gmail.com");

            //create Authenticator object to pass in Session.getInstance argument
            Authenticator auth = new Authenticator() {
                //override the getPasswordAuthentication method
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(fromEmail, password);
                }
            };
            Session session = Session.getInstance(props, auth);

            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress(fromEmail));
            message.addRecipient(Message.RecipientType.TO, new InternetAddress(toEmail));

            System.out.println("Mail Check 2");

            message.setSubject(subject);


            // IF YOU WANT TO SEND HTML, USE THIS LINE OF CODE INSTEAD:
            message.setContent(html, "text/html; charset=utf-8");
            //message.setText("");


            Transport.send(message);
        }catch(Exception ex){
            System.out.println("Mail fail");
            System.out.println(ex);
        }
    }


    public static String createNotificationHTML(String name, int project_id, String new_status) {
        return header + "<h3>Hi, " + name + "</h3>\n" +
                "<p class=\"lead\">Project #" + project_id + " has advanced to a new step, " + new_status + "</p>\n" +
                "<!-- Callout Panel -->\n" +
                "<p class=\"callout\">\n" +
                "Check out the <a href=\"http://localhost:8080/project?id=" + project_id + "\">changes here</a>\n" +
                "</p><!-- /Callout Panel -->\n" + footer;
    }

    public static String createWelcomeHTML(String name, String link) {
        return header + "<h3>Welcome to Dell's Campaign Management System, " + name + "!</h3>\n" +
                "<p class=\"lead\">To set get started, set your password and you will be logged in immediately</p>\n" +
                "<p>If you need any help with the system, visit our <a href=\"localhost:8080/help\">help section</a></p>\n" +
                "<p class=\"callout\">\n" +
                "<a href=\"" + link + "\">Set your password</a>\n" +
                "</p><!-- /Callout Panel -->\n" + footer;
    }

    private static String header = "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\"
\"http://www.w3.org/TR/REC-html40/loose.dtd\">\n" +
            "<html style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n"
+
            "<head></head>\n" +
            "<body style=\"-webkit-font-smoothing: antialiased; -webkit-text-size-adjust: none; font-family: 'Helvetica Neue',
'Helvetica', Helvetica, Arial, sans-serif; height: 100%; margin: 0; padding: 0; width: 100% !important\">\n" +
            "<style type=\"text/css\">\n" +
            "img {\n" +
            "max-width: 100%;\n" +
            "}\n" +
            "body {\n" +
            "-webkit-font-smoothing: antialiased; -webkit-text-size-adjust: none; width: 100% !important; height: 100%;\n" +
            "}\n" +
            "@media only screen and (max-width: 600px) {\n" +
            "  a[class=\"btn\"] {\n" +
            "    display: block !important; margin-bottom: 10px !important; background-image: none !important; margin-right:
0 !important;\n" +
            "  }\n" +
            "  div[class=\"column\"] {\n" +
            "    width: auto !important; float: none !important;\n" +
            "  }\n" +
            "  table.social div[class=\"column\"] {\n" +
            "    width: auto !important;\n" +
            "  }\n" +
            "}\n" +
            "</style>\n" +
            "<div class=\"body\" bgcolor=\"#FFFFFF\" style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial,
sans-serif; margin: 0; padding: 0\">\n" +
            "<!-- HEADER -->\n" +
            "<table class=\"head-wrap\" bgcolor=\"#0085C3\" style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica,
Arial, sans-serif; margin: 0; padding: 0; width: 100%\"><tr style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica,
Arial, sans-serif; margin: 0; padding: 0\">\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "<td class=\"header container\" style=\"clear: both !important; display: block !important; font-family: 'Helvetica
Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0 auto; max-width: 600px !important; padding: 0\">\n" +
            "<div class=\"content\" style=\"display: block; font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-
serif; margin: 0 auto; max-width: 600px; padding: 15px\">\n" +
            "<table bgcolor=\"#0085C3\" style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
margin: 0; padding: 0; width: 100%\"><tr style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
margin: 0; padding: 0\">\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\"><img
src=\"http://f.cl.ly/items/2T3e1p3R2j21261Q3Q3V/dell-logo.png\" style=\"font-family: 'Helvetica Neue', 'Helvetica',
Helvetica, Arial, sans-serif; margin: 0; max-width: 100%; padding: 0; width: 50px\"></td>\n" +
            "<td align=\"right\" style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0;
padding: 0\"><h6 class=\"collapse\" style=\"color: white; font-family: 'HelveticaNeue-Light', 'Helvetica Neue Light',
'Helvetica Neue', Helvetica, Arial, 'Lucida Grande', sans-serif; font-size: 14px; font-weight: normal; line-height: 1.1; margin:
0; padding: 0; text-transform: uppercase\">Campaign management system</h6></td>\n" +
            "</tr></table>\n" +
            "</div>\n" +
            "</td>\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "</tr></table>\n" +
            "<!-- /HEADER --><!-- BODY --><table class=\"body-wrap\" style=\"font-family: 'Helvetica Neue', 'Helvetica',
Helvetica, Arial, sans-serif; margin: 0; padding: 0; width: 100%\"><tr style=\"font-family: 'Helvetica Neue', 'Helvetica',
Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "<td class=\"container\" bgcolor=\"#FFFFFF\" style=\"clear: both !important; display: block !important; font-
family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0 auto; max-width: 600px !important; padding:
0\">\n" +
            "\n" +
            "<div class=\"content\" style=\"display: block; font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-
serif; margin: 0 auto; max-width: 600px; padding: 15px\">\n" +
            "<table style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0; width:
100%\"><tr style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n";

    private static String footer =  "</tr></table>\n" +
            "</div>\n" +
            "<!-- /content -->\n" +
            "\n" +
            "</td>\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "</tr></table>\n" +
            "<!-- /BODY --><!-- FOOTER --><table class=\"footer-wrap\" style=\"clear: both !important; font-family:
'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0; width: 100%\"><tr style=\"font-family:
'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "<td class=\"container\" style=\"clear: both !important; display: block !important; font-family: 'Helvetica Neue',
'Helvetica', Helvetica, Arial, sans-serif; margin: 0 auto; max-width: 600px !important; padding: 0\">\n" +
            "\n" +
            "<!-- content -->\n" +
            "<div class=\"content\" style=\"display: block; font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-
serif; margin: 0 auto; max-width: 600px; padding: 15px\">\n" +
            "<table style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0; width:
100%\"><tr style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding: 0\">\n" +
            "<td align=\"center\" style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0;
padding: 0\">\n" +
            "<p style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; font-size: 14px; font-weight:
normal; line-height: 1.6; margin: 0 0 10px; padding: 0\">\n" +
            "<a href=\"#\" style=\"color: #2BA6CB; font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
margin: 0; padding: 0\">Campaign System</a> |\n" +
            "<a href=\"#\" style=\"color: #2BA6CB; font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
margin: 0; padding: 0\">Dell homepage</a>\n" +
            "</p>\n" +
            "</td>\n" +
            "</tr></table>\n" +
            "</div>\n" +
            "<!-- /content -->\n" +
            "\n" +
            "</td>\n" +
            "<td style=\"font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif; margin: 0; padding:
0\"></td>\n" +
            "</tr></table>\n" +
            "<!-- /FOOTER -->\n" +
            "</div>\n" +
            "</body>\n" +
            "</html>";
  }
```

Poe

```
package Domain;

import javafx.scene.web.HTMLEditor;

import java.io.File;
```

```java
import java.net.URLEncoder;
import java.sql.Time;
import java.sql.Timestamp;


public class Poe {

    int id;
    int proj_id;
    String filename;
    int user_id;
    Timestamp date;
    Timestamp deletion_date;
    String filetype;
    int uploaded_on_stage;

    long f_date;
    long f_deletion_date;

    public Poe(int id, int proj_id, String filename, int user_id, Timestamp date, String filetype, Timestamp deletion_date, int
uploaded_on_stage) {
        this.id = id;
        this.proj_id = proj_id;
        this.filename = filename;
        this.user_id = user_id;
        this.date = date;
        this.filetype = filetype;
        this.deletion_date = deletion_date;
        this.uploaded_on_stage = uploaded_on_stage;

        if(date != null) f_date = date.getTime();
        if(deletion_date != null) f_deletion_date = deletion_date.getTime();
    }

    public int getId() {
        return id;
    }

    public int getProj_id() {
        return proj_id;
    }

    public String getFilename() {
        return URLEncoder.encode(filename);
    }

    public int getUser_id() {
        return user_id;
    }

    public Timestamp getDate() {
        return date;
    }

    public String getFiletype() {
        return filetype;
    }

    public long getF_date() {
        return f_date;
    }

    public long getF_deletion_date() {
        return f_deletion_date;
    }
    public int getUploaded_on_stage() {
        return uploaded_on_stage;
    }

    public String getFilePath() {
        return System.getenv("POE_FOLDER") + File.separator + proj_id + File.separator + filename;
    }

}
```

Project

```java
package Domain;

import java.sql.Timestamp;

public class Project {

    public int id;
    public Timestamp start_time;
    public Timestamp end_time;
    public int company_id;
    public int owner_id;
    public String status;
    public double budget;
    public String body;
    public Timestamp execution_date;
    public Timestamp last_change_admin;
    public Timestamp last_change_partner;
    public boolean unread_admin;
    public boolean unread_partner;
    public String notification;
    public String type;

    public Project() {

    };

    public Project(int id, Timestamp start_time, Timestamp end_time, int company_id, int owner_id, String status, double
budget, String body, Timestamp execution_date, Timestamp last_change_admin, Timestamp last_change_partner, boolean
unread_admin, boolean unread_partner, String notification, String type) {

        this.id = id;
        this.start_time = start_time;
        this.end_time = end_time;
        this.company_id = company_id;
        this.owner_id = owner_id;
        this.status = status;
        this.budget = budget;
        this.body = body;
        this.execution_date = execution_date;
        this.last_change_admin = last_change_admin;
        this.last_change_partner = last_change_partner;
        this.unread_admin = unread_admin;
        this.unread_partner = unread_partner;
        this.notification = notification;
        this.type = type;

    }
    public boolean isUnread_partner() {
        return unread_partner;
    }

    public int getId() {
        return id;
    }
    public Timestamp getStart_time() {
        return start_time;
    }

    public Timestamp getEnd_time() {
        return end_time;
    }

    public int getCompany_id() {
        return company_id;
    }

    public int getOwner_id() {
        return owner_id;
    }

    public String getStatus() {
        return status;
    }

    public double getBudget() {
        return budget;
    }

    public String getBody() {
        return body;
    }

    public Timestamp getExecution_date() {
        return execution_date;
    }

    public Timestamp getLast_change_admin() {
        return last_change_admin;
    }

    public Timestamp getLast_change_partner() {
        return last_change_partner;
    }

    public boolean isUnread_admin() { return unread_admin; }

    public String getNotification() { return notification; }

    public String getType() { return type; }


    @Override
    public String toString() {
        return "" + id + ": " + body;
    }
}
```

Result

```
package Domain;

public class Result {

    public String type;
    public int id;
    public String body;
    public int rating;

    public Result(String type, int id, String body, int rating) {
        this.type = type;
        this.id = id;
        this.body = body;
        this.rating = rating;
    }

    public String getType() {
        return type;
    }

    public int getRating() {
        return rating;
    }

    public int getId() {
        return id;
    }

    public String getBody() {
        return body;
    }
}
```

ResultContainer

```
package Domain;

import java.util.ArrayList;

public class ResultsContainer {

    public String type;
    public ArrayList<Result> container;

    public ResultsContainer(String type) {
        this.type = type;
        container = new ArrayList<>();
    }

    public String getType() {
        return type;
    }

    public ArrayList<Result> getContainer() {
        return container;
    }
}
```

Stage

```
package Domain;

import java.sql.Timestamp;
import java.util.Comparator;

public class Stage {

    public int id;

    public int user_id;
    public int project_id;
    public Long date;
    public String type;

    public User user;

    public Stage(int id, int user_id, int project_id, Timestamp date, String type) {
        this.id = id;
        this.user_id = user_id;
        this.project_id = project_id;
        if(date != null) this.date = date.getTime();
        this.type = type;
    }

    public static final Comparator<Stage> TIME = (Stage o1, Stage o2) -> o1.date.compareTo(o2.date);

    public int getId() {
        return id;
    }

    public int getUser_id() {
        return user_id;
    }

    public int getProject_id() {
        return project_id;
    }

    public Long getDate() {
        return date;
    }

    public String getType() {
        return type;
    }

    public User getUser() {
        return user;
    }

}
```

User

```
package Domain;

public class User {

    public int id;
    public String name;
    public String password;
    public String email;
    public int company_id;
    public boolean deleted;
    public Company company;

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getPassword() {
        return password;
    }
    public String getEmail() {
        return email;
    }

    public int getCompany_id() { return company_id; }

    public boolean getDeleted() { return deleted; }

    public Company getCompany() { return company; }

    public User(int id1, String name1, String email1, String password1, int company_id1, boolean deleted) {

        this.id = id1;
        this.name = name1;
        this.password = password1;
        this.email = email1;
        this.company_id = company_id1;
        this.deleted = deleted;

    }

    public String toString()
    {
        return "IM HERE";
    }

}
```

PresentationServlet

```
package Presentation;

import Domain.Budget;
import Domain.Company;
import Domain.Controller;
import Domain.User;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.*;
import java.io.*;
import java.net.URLDecoder;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Calendar;

import static org.apache.commons.lang3.StringEscapeUtils.*;
import static org.apache.commons.lang3.StringUtils.*;

@MultipartConfig
public class PresentationServlet extends HttpServlet {

    protected void process(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
        Controller cont = AssignController(request);

        System.out.println(request.getRequestURI());

        // if logged in
        Object userObj = request.getSession().getAttribute("User");
        if (userObj != null) {
            request.setAttribute("User", userObj); // passing user object to request
        String userPath = request.getServletPath();

        getActiveBudget(request, response, cont);

        if(userPath.indexOf("/resources/") == 0) {
            serveResource(request, response, cont);
        } else {

            switch (userPath) {
                case "/dashboard":
                    getDashboard(request, response, cont);
                    break;
                case "/partners":
                    getPartners(request, response, cont);
                    break;
                case "/partner":
                    getPartnerView(request, response, cont);
                    break;
                case "/users":
                    getUsers(request, response, cont);
                    break;
                case "/user":
                    getUserView(request, response, cont);
                    break;
                case "/budgets":
                    getBudgets(request, response, cont);
                    break;
                case "/edit-budget":
                    getEditBudget(request, response, cont);
                    break;
                case "/project-request":
                    request.getRequestDispatcher("/WEB-INF/view/createproject.jsp").forward(request, response);
                    break;
                case "/project":
                    getProjectView(request, response, cont);
                    break;
                case "/create-company":
                    getCreateCompanyView(request, response, cont);
                    break;
                case "/getCompanyNames":
                    getCompanyNames(request, response, cont);
                    break;
                case "/statistics":
                    getStatisticsView(request, response, cont);
                    break;
                case "/getStatuses":
                    getDistinctStatuses(request, response, cont);
                    break;
                case "/getTypes":
                    getDistinctTypes(request, response, cont);
                    break;
                case "/create-user":
                    getCreateUserView(request, response, cont);
                    break;
                case "/create-budget":
                    request.getRequestDispatcher("/WEB-INF/view/create-budget.jsp").forward(request, response);
                    break;
                case "/logout":
                    logout(request, response, cont);
                    break;
                default:
                    response.sendRedirect("/dashboard");
                    break;
            }
        }
    } else {
        String userpath = request.getRequestURI();
```

```
            System.out.println(userpath);
            if(userpath.equals("/login"))
                request.getRequestDispatcher("/WEB-INF/view/login.jsp").forward(request, response);
            else if(userpath.equals("/reset-password"))
                getCreatePasswordView(request, response, cont);
            else
                response.sendRedirect("/login");
        }
    }


    protected void processPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        Controller cont = AssignController(request);
        String path = request.getRequestURI();

        System.out.println(path);

        //request.setAttribute("error", false);

        Object userObj = request.getSession().getAttribute("User");
        if (userObj != null)
            request.setAttribute("User", userObj); // passing user object to request

        getActiveBudget(request, response, cont);

        switch (path) {
            case "/login":
                login(request, response, cont);
                break;
            case "/api/getUserById":
                getUserById(request, response, cont);
                break;
            case "/api/postMessage":
                postMessage(request, response, cont);
                break;
            case "/api/getProjectsByState":
                getProjectsByState(request, response, cont);
                break;
            case "/api/changeProjectStatus":
                changeProjectStatus(request, response, cont);
                break;
            case "/api/createCompany":
                createCompany(request, response, cont);
                break;
            case "/api/createUser":
                createUser(request, response, cont);
                break;
            case "/project-request":
                createProjectRequest(request, response, cont);
                break;
            case "/uploadFile":
                createPoe(request, response, cont);
                break;
            case "/downloadFile":
                getPoes(request, response, cont);
                break;
            case "/api/deleteFile":
                deletePoe(request, response, cont);
                break;
            case "/createUser":
                createUser(request, response, cont);
                break;
            case "/reset-password":
                createPassword(request, response, cont);
                break;
            case "/createBudget":
                createBudget(request, response, cont);
                break;
            case "/modifyBudget":
                modifyBudget(request, response, cont);
                break;
            case "/markUserDeleted":
                deleteUser(request, response, cont);
                break;
            default:
                getDashboard(request, response, cont);
        }
    }

    private Controller AssignController(HttpServletRequest request) {

        HttpSession sessionObj = request.getSession();
        Controller cont = (Controller) sessionObj.getAttribute("Controller"); // becomes null initially
        if (cont == null)
        {
            // Start new session
            // Not using singleton; each user will be given their own controller for use throughout their session
            cont = new Controller();
            sessionObj.setAttribute("Controller", cont);
        } else
        {
            // Continue ongoing session
            cont = (Controller) sessionObj.getAttribute("Controller");
        }
        return cont;

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
        processPost(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
        process(request, response);
    }

    void login(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        Object user = cont.login(getString("email", request), getString("password", request));
        if (user != null) {
            request.getSession().setAttribute("User", user);
            response.sendRedirect("/dashboard");
        } else {
            request.setAttribute("message", "Incorrect login");
            request.getRequestDispatcher("/WEB-INF/view/login.jsp").forward(request, response);
        }

    }

    void getDashboard(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("getDashboard");

        User user = (User) request.getAttribute("User");

        //if search
        if(getLazyString("q", request) != null) {
            String q = getLazyString("q", request);
            request.setAttribute("results", cont.search(q, user.getCompany_id()));
        } else {
            if(getLazyString("state", request) != null)
                request.setAttribute("projects", cont.getProjectsByState(getLazyString("state", request), user.getCompany_id()));
            else if (getLazyString("type", request) != null)
                request.setAttribute("projects", cont.getProjectsByType(getLazyString("type", request), user.getCompany_id()));
            else if (getLazyString("company", request) != null)
                request.setAttribute("projects", cont.getProjectsByCompanyName(getLazyString("company", request),
user.getCompany_id()));
            else
                request.setAttribute("projects", cont.getProjectsByState("waitingForAction", user.getCompany_id()));
        }

        request.setAttribute("statusCount", cont.getStatusCounts(user.getCompany_id()));

        request.getRequestDispatcher("/WEB-INF/view/index.jsp").forward(request, response);
    }

    void getPartners(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("getPartners");
        User user = (User) request.getAttribute("User");
        if (user.getCompany_id() != 1) {
            response.sendRedirect("/dashboard");
        } else {
            request.setAttribute("partners", cont.getCompanies());
            request.getRequestDispatcher("/WEB-INF/view/partners.jsp").forward(request, response);
        }
    }

    void getUsers(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("getUsers");
        User user = (User) request.getAttribute("User");
        if (user.getCompany_id() != 1) {
            response.sendRedirect("/dashboard");
        } else {
            request.setAttribute("users", cont.getUsers());
            request.getRequestDispatcher("/WEB-INF/view/users.jsp").forward(request, response);
        }
    }

    void getBudgets(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("getBudgets");
        User user = (User) request.getAttribute("User");
        if (user.getCompany_id() != 1) {
            response.sendRedirect("/dashboard");
        } else {
            request.setAttribute("budgets", cont.getAllBudgets());
            request.getRequestDispatcher("/WEB-INF/view/budgets.jsp").forward(request, response);
        }
    }

    void getEditBudget(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        request.setAttribute("initialbudget", getString("initialbudget", request));
        int year = getInt("year", request);
        request.setAttribute("year", year);
        int quarter = getInt("quarter", request);
        request.setAttribute("quarter", quarter);

        request.getRequestDispatcher("/WEB-INF/view/edit-budget.jsp").forward(request, response);
    }

    void getProjectView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        System.out.println("getProjectView");
        User user = (User) request.getAttribute("User");
        int projId = getInt("id", request);
```

```java
        Object proj = cont.getProjectById(projId, user.getCompany_id());
        if(proj == null) {
            error("Project not found", request, response, cont);
            return;
        }
        request.setAttribute("project", proj);
        request.setAttribute("messages", cont.getMessagesByProjectId(projId));
        request.setAttribute("stages", cont.getStagesByProjectId(projId));
        request.setAttribute("poes", cont.getPoe(projId));

        request.getRequestDispatcher("/WEB-INF/view/project.jsp").forward(request, response);
    }

    void getPartnerView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        User user = (User) request.getAttribute("User");
        if (user.getCompany_id() != 1) {
            response.sendRedirect("/dashboard");
        } else {
            int partnerId = getInt("id", request);
            request.setAttribute("partner", cont.getCompanyById(partnerId));
            request.setAttribute("users", cont.getUsersByCompanyId(partnerId));
            request.setAttribute("projects", cont.getProjectsByCompanyId(partnerId));

            request.getRequestDispatcher("/WEB-INF/view/partner.jsp").forward(request, response);
        }
    }

    void getUserView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("getUserView");
        User user = (User) request.getAttribute("User");
        if (user.getCompany_id() != 1) {
            response.sendRedirect("/dashboard");
        } else {
            int userId = getInt("id", request);
            User tempUser = cont.getUserById(userId);
            request.setAttribute("user", tempUser);
            try {
                Company company = cont.getCompanyById(tempUser.getCompany_id());
                request.setAttribute("partner", company);
                request.setAttribute("projects", cont.getProjectsByUserId(userId));
            } catch (NullPointerException e) {
                getUsers(request, response, cont);
                return;
            }

            request.getRequestDispatcher("/WEB-INF/view/user.jsp").forward(request, response);
        }
    }

    void postMessage(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        System.out.println("postMessage");

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        int userId = getInt("userId", request);
        int companyId = getInt("companyId", request);
        int projectId = getInt("projectId", request);

        String body = getString("body", request);

        if(request.getAttribute("error") != null)
            out.println("Error - Invalid input");
        else
```

```java
            out.println(cont.postMessage(userId, projectId, body, companyId));
    }

    void getUserById (HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int user_id = getInt("user_id", request);
        User user = cont.getUserById(user_id);
        String user_info = user.toString();
        request.setAttribute("userInfo", user_info);
        request.getRequestDispatcher("index.jsp").forward(request, response);
    }

    void createProjectRequest(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        String project_body = getString("body", request);
        int budget = getInt("budget", request);
        String project_type = getString("type", request);


        int execution_year = getInt("execution_year", request);
        int execution_month = getInt("execution_month", request);
        int execution_day = getInt("execution_day", request);

        Timestamp execution_time;

        if (execution_day == 0) {
            execution_time = Timestamp.valueOf(execution_year + "-" + execution_month + "-01" + " 00:00:01");
        } else {
            execution_time = Timestamp.valueOf(execution_year + "-" + execution_month + "-" + execution_day + "
00:00:00");
        }
        System.out.println("error?: " + request.getAttribute("error"));
        if(request.getAttribute("error") != null) {
            ArrayList<String[]> formData = new ArrayList<>();
            formData.add(new String[]{"body", project_body.replaceAll("\r\n", "\\\\n")});
            formData.add(new String[] {"budget", request.getParameter("budget")});
            formData.add(new String[] {"type", project_type});
            formData.add(new String[] {"execution_year", String.valueOf(execution_year)});
            formData.add(new String[] {"execution_month", String.valueOf(execution_month)});
            formData.add(new String[] {"execution_day", String.valueOf(execution_day)});

            request.getSession().setAttribute("formData", formData);
            response.sendRedirect("/project-request");
        } else {
            User user = (User) request.getAttribute("User");

            int projectId = cont.createProjectRequest(budget, project_body, user, project_type, execution_time);

            if (projectId != 0) {
                response.sendRedirect("/project?id=" + projectId);
            } else {
                System.out.println("Project ID is 0!");
            }
        }
    }

    void getProjectsByState(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        User user = (User) request.getAttribute("User");
        request.setAttribute("projects", cont.getProjectsByState(getString("state", request), user.getCompany_id()));
        request.getRequestDispatcher("index.jsp").forward(request, response);
    }



    void changeProjectStatus(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        int projectId = getInt("projectId", request);
        String currentType = getString("currentType", request);
        String answer = getString("answer", request);
        User u = (User) request.getAttribute("User");
        int companyId = u.getCompany_id();
        int userId = u.getId();

        if(request.getAttribute("error") == null)
            cont.changeProjectStatus(projectId, currentType, answer, companyId, userId);
        response.sendRedirect("/project?id=" + projectId);
    }

    void logout(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
            response.sendRedirect("/login");
        }
    }


    void createPoe(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        Part file = request.getPart("file");
        User u = (User) request.getAttribute("User");
        int project_id = getInt("proj_id", request);
        int user_id = u.getId();
        int stage = -1;
        if(getString("stage", request) != null)
            stage = getInt("stage", request);
        if(cont.addPoeFile(project_id, file, user_id, stage)) {
            response.sendRedirect("/project?id=" + project_id);
        }
    }


    void deletePoe(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int projectId = getInt("projectId", request);
        String fileName = getString("fileName", request);
        int fileId = getInt("fileId", request);
        boolean deleteFile = Boolean.parseBoolean(request.getParameter("deleteFile"));

        cont.deleteFile(fileName, projectId, fileId, deleteFile);

        response.sendRedirect("/project?id=" + projectId);
    }


    void getCreateCompanyView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        ArrayList<Company> companies = cont.getCompanies();
        request.setAttribute("companies", companies);
        request.getRequestDispatcher("/WEB-INF/view/create-company.jsp").forward(request, response);
    }

    void getCreateUserView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        ArrayList<Company> companies = cont.getCompanies();
        request.setAttribute("companies", companies);
        request.setAttribute("partnerName", request.getParameter("partnerName"));
        request.getRequestDispatcher("/WEB-INF/view/create-user.jsp").forward(request, response);
    }

    void createCompany(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
```

```java
        String company_name = getString("companyName", request);
        String country_code = getString("countryCode", request);
        Part logo = null;
        if(request.getParameter("logo") != null)
            logo = request.getPart("logo");
        String logo_url = getLazyString("logoUrl", request);


        if(request.getAttribute("error") == null) {
            int company_id = cont.createCompany(company_name, country_code, logo, logo_url);
            if (company_id > 0) { //if success
                //request.setAttribute("message", "Create the first user for this company by clicking 'Add user'");
                setMessage("Create the first user for this company by clicking 'Add user'", request);
                response.sendRedirect("/partner?id=" + company_id);
            } else {
                request.setAttribute("error", "Something went wrong, try again");
                request.getRequestDispatcher("/WEB-INF/view/create-company.jsp").forward(request, response);
            }
        } else
            response.sendRedirect("/create-company");
    }

    void getPoes(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int project_id = getInt("proj_id", request);
        String filename = URLDecoder.decode(getString("filename", request));

        response.setContentType("application/octet-stream");
        response.setHeader("Content-Disposition",
                "attachment;filename=" + filename);

        // testing first poe
        String path = System.getenv("POE_FOLDER") + File.separator + project_id + File.separator + filename;

        File file = new File(path);
        FileInputStream fileIn = new FileInputStream(file);
        ServletOutputStream out = response.getOutputStream();

        byte[] outputByte = new byte[4096];

        while(fileIn.read(outputByte, 0, 4096) != -1)
        {
            out.write(outputByte, 0, 4096);
        }
        fileIn.close();
        out.flush();
        out.close();

        response.sendRedirect("/");
    }

    void serveResource(HttpServletRequest request, HttpServletResponse response, Controller cont) {
        try {
            String userpath = request.getServletPath();
            boolean download = Boolean.parseBoolean(request.getParameter("download"));

            String[] path = userpath.split("/");
            String filename = System.getenv("POE_FOLDER");

            for (int i=2; i < path.length; i++) {
                filename+= File.separator + path[i];
            }

            //String filename = System.getenv("POE_FOLDER") + File.separator + userpath.split("/")[2] + File.separator +
userpath.split("/")[3];
            File file = new File(filename);
            if(download) {
                response.setContentType("application/force-download");
                //response.setContentLength(-1);
                response.setHeader("Content-Transfer-Encoding", "binary");
                response.setHeader("Content-Disposition","attachment; filename=\"" + file.getName() + "\"");
            } else {
                ServletContext cntx= getServletContext();

                // retrieve mimeType dynamically
                String mime = cntx.getMimeType(filename);
                if (mime == null) {
                    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
                    return;
                }
                response.setContentType(mime);
            }


            response.setContentLength((int)file.length());

            FileInputStream in = new FileInputStream(file);
            OutputStream out = response.getOutputStream();

            // Copy the contents of the file to the output stream
            byte[] buf = new byte[1024];
            int count = 0;
            while ((count = in.read(buf)) >= 0) {
                out.write(buf, 0, count);
            }
            out.close();
            in.close();
        } catch (Exception e) {};

    }

    void createUser(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        String name = getString("userName", request);
        String email = getString("userEmail", request);
        int company_id = getInt("selectedCompany", request);
        if(request.getAttribute("error") == null) {
            int id = cont.createUser(name, email, company_id);
            if (id == -1) {
                setError("Something went wrong.", null, request);
                response.sendRedirect("/create-user");
            } else
                response.sendRedirect("/user?id=" + id);
        } else {
            ArrayList<String[]> formData = new ArrayList<>();
            formData.add(new String[] {"userName", name});
            formData.add(new String[] {"userEmail", email});
            formData.add(new String[] {"selectedCompany", String.valueOf(company_id)});
            request.getSession().setAttribute("formData", formData);
            response.sendRedirect("/create-user");
        }

    }

    void createBudget(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int year = getInt("year", request);
        int quarter = getInt("quarter", request);
        int budget = getInt("initial_budget", request);

        if(request.getAttribute("error") == null) {
            if (cont.addBudget(year, quarter, budget)) {
                cont.sendEmail("noobglivestream@gmail.com", "budget created", "hope this works!");
                response.sendRedirect("/budgets");
            } else {
                request.setAttribute("errorMes", "Quarter already exists, consider modifying the current budget or creating a new
one.");
                response.sendRedirect("/budget_view");
            }
        } else
            response.sendRedirect("/create-budget");
    }

    void getStatisticsView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        request.setAttribute("statistics", cont.getStatistics(getLazyString("quarter", request)));
        request.setAttribute("budgets", cont.getAllBudgets());
        request.getRequestDispatcher("/WEB-INF/view/statistics.jsp").forward(request, response);
    }


    void getDistinctStatuses(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        out.print(cont.getDistinctStatuses(getString("query", request)));
    }
    void getDistinctTypes(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        User user = (User) request.getAttribute("User");
        out.print(cont.getDistinctTypes(getString("query", request), user.getCompany_id()));
    }
    void getCompanyNames(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        User user = (User) request.getAttribute("User");
        out.print(cont.getCompanyNames(getString("query", request), user.getCompany_id()));
    }


    void error(String error, HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        request.setAttribute("error", error);
        getDashboard(request, response, cont);
    }

    void getActiveBudget(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        int currentMonth = Calendar.getInstance().get(Calendar.MONTH);
        System.out.println(currentMonth);
        int currentYear = Calendar.getInstance().get(Calendar.YEAR);
        int currentQuarter = (currentMonth / 3 ) + 1;

        request.setAttribute("activeBudget", cont.getActiveBudget(currentYear, currentQuarter));



    }

    void modifyBudget(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int newBudget = getInt("newBudget", request);
        int year = getInt("year", request);
        int quarter = getInt("quarter", request);


        if(request.getAttribute("error") == null) {
            cont.modifyBudget(newBudget, year, quarter);
        }
        response.sendRedirect("/budget_view");
    }

    void getCreatePasswordView(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        String nonce = getString("n", request);
        int userId = cont.getUserIdByNonce(nonce);
        request.setAttribute("userId", userId);

        request.getRequestDispatcher("/WEB-INF/view/reset.jsp").forward(request, response);
    }

    void createPassword(HttpServletRequest request, HttpServletResponse response, Controller cont) throws
ServletException, IOException {
        String nonce = getString("nonce", request);
        String password = getString("pw", request);
        int id = getInt("user_id", request);

        if(request.getAttribute("error") == null) {

            if (cont.createPassword(id, password, nonce)) {
                request.getSession().setAttribute("User", cont.getUserById(id));
                response.sendRedirect("/dashboard");
            } else {
                request.setAttribute("error", "Something went wrong, try recovering password again");
                request.getRequestDispatcher("/WEB-INF/view/reset.jsp").forward(request, response);
            }
        } else
            response.sendRedirect("/reset-password");
    }

    String getString(String p, HttpServletRequest request) {
        String s = request.getParameter(p);
        if(s == null || s.length() == 0 || s.equals("")) {
            setError("Empty field", p, request);
            request.setAttribute("error", true);
        }
        s = escapeHtml4(s);
        return s;
    }
    String getLazyString(String p, HttpServletRequest request) {
        return escapeHtml4(request.getParameter(p));
    }
    int getInt(String p, HttpServletRequest request) {
        String s = request.getParameter(p);
        if(s == null || s.length() == 0 || s.equals("")) {
            request.setAttribute("error", true);
            setError("Missing field", p, request);
        }
        int i = -1;
        if(isNumeric(s))
            i = Integer.parseInt(s);
        else {
            request.setAttribute("error", true);
            setError("Enter only numbers ",p , request);
        }

        return i;
    }

    void setMessage(String message, HttpServletRequest request) {
        request.getSession().setAttribute("message", message);
    }
    void setError(String error, String field, HttpServletRequest request) {
        String e = (String) (request.getSession().getAttribute("errorMessage"));
        if(e == null)
            request.getSession().setAttribute("errorMessage", error + "|" + field);
        else
            request.getSession().setAttribute("errorMessage", e + "," + field);
    }

    void deleteUser(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException,
IOException {
        int viewedUser = Integer.parseInt(request.getParameter("viewedUser"));

        cont.markUserDeleted(viewedUser);
        response.sendRedirect("/users");
    }

}
```

MessageItemByIndex

```
<div class="item message <c:if test="${messages.get(messageIndex).getUser().getId() == User.getId()}">pull-right</c:if>">
    <span class="user-data">
        <c:out value="${messages.get(messageIndex).getUser().getName()}"></c:out> -
        <c:out value="${messages.get(messageIndex).getCompany().getName()}"></c:out>
    </span>
    <span class="date isDate">
        <c:out value="${messages.get(messageIndex).getCreation_date_millis()}"></c:out>
    </span>
    <div class="inner-bubble">
        <c:out value="${messages.get(messageIndex).getBody()}" escapeXml="false"></c:out>
    </div>
</div>
```

StageItemByIndex

```
<%@ taglib uri='http://java.sun.com/jsp/jstl/core' prefix='c'%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<c:set var="dellAndLatest" value="${User.getCompany_id() == 1 && lastStage}"></c:set>
<c:set var="partnerAndLatest" value="${User.getCompany_id() != 1 && lastStage}"></c:set>

<div class="item<c:if test='${stages.get(stageIndex).getUser().getId() == User.getId()}'> pull-right</c:if>">
    <span class="user-data"><c:out value="${stages.get(stageIndex).getUser().getName()}"></c:out> - <c:out
value="${stages.get(stageIndex).getUser().getCompany().getName()}"></c:out></span>
    <span class="date isDate"><c:out value="${stages.get(stageIndex).getDate()}"></c:out></span>
    <div class="bubble">
        <div class="inner-bubble <c:if test="${stages.get(stageIndex).getType() == 'Project Approved'}">approved</c:if> <c:if
test="${stages.get(stageIndex).getType() == 'Project Rejected' || stages.get(stageIndex).getType() == 'Claim
Rejected'}">rejected</c:if>">

        <c:if test="${stages.get(stageIndex).getType() == 'Waiting Project Verification'}">
            <c:set var="balance" value="${project.getBudget()}" />
            <fmt:formatNumber var="i" type="number" value="${balance}" />
            <h3><c:out value="${project.getCompanyName()}"></c:out> <strong>is requesting</strong> <c:out
value="${i}"></c:out>&#8364 <strong>for a</strong> <c:out value="${project.getType()}"></c:out></h3>
            <c:out value="${project.getBody()}" escapeXml="false"></c:out>
            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">Review the project; approve if satisfactory or reject if not.</p>
            </c:if><c:if test="${partnerAndLatest}">
                <p class="status-message">Dell will now review your request.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Project Rejected'}">
            <h3>Project has been rejected</h3>
            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">Add a comment to explain your decision of rejecting the project.</p>
            </c:if>
            <c:if test="${partnerAndLatest}">
                <p class="status-message">Dell has rejected your request. Change your request to match Dell�s comment.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Project Approved'}">
            <h3>Project has been Approved</h3>
            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">You have now approved the project and partner is working on executing the project. A
claim request will be made once executed.</p>
            </c:if>
            <c:if test="${partnerAndLatest}">
                <p class="status-message">You are now allowed to execute the project. When executed, submit a claim with
proper proof of execution.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Waiting Claim Verification'}">
            <h3>Waiting claim verification</h3>
            <c:forEach items="${poes}" var="poe" varStatus="ite" >
                <c:if test="${poe.getF_date() < stages.get(stageIndex).getDate() || (poe.getUploaded_on_stage() ==
stages.get(stageIndex).getId()) }">
                    <c:if test="${poe.getF_deletion_date() == 0 || (poe.getF_deletion_date() != 0 && poe.getF_deletion_date() >
stages.get(stageIndex).getDate() && stageIndex + 1 != stages.size())}">
                        <div class="proof-container <c:if test="${poe.getF_date() > stages.get(stageIndex - 1).getDate() &&
poe.getF_date() < stages.get(stageIndex).getDate()}"> new</c:if>">
                            <c:choose>
                                <c:when test="${poe.getFiletype() == 'jpg' || poe.getFiletype() == 'png' || poe.getFiletype() == 'jpeg' ||
poe.getFiletype() == 'gif' || poe.getFiletype() == 'bmp'}">
                                    <div class="proof" style="background-image: url(/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out value='${poe.getFilename()}'></c:out>)">
                                        <a class="fancybox" rel="<c:out value='${poe.getProj_id()}'></c:out>" href="/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out value='${poe.getFilename()}'></c:out>"><div class="view-
image"></div></a>
                                        <div class="download-file"><a href="/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out
value='${poe.getFilename()}'></c:out>?download=true">Download</a></div>
                                    </div>
                                </c:when>
                                <c:otherwise>
                                    <div class="proof
                                        <c:choose>
                                            <c:when test="${poe.getFiletype() == 'xlsx' || poe.getFiletype() == 'xls' || poe.getFiletype() ==
'numbers' || poe.getFiletype() == 'xml'}">
                                                excel
                                            </c:when>
                                            <c:when test="${poe.getFiletype() == 'zip' || poe.getFiletype() == 'rar' || poe.getFiletype() ==
'tar' || poe.getFiletype() == 'dmg'}">
                                                archive
                                            </c:when>
                                            <c:when test="${poe.getFiletype() == 'mp3' || poe.getFiletype() == 'flac' || poe.getFiletype()
== 'm4a' || poe.getFiletype() == 'wav' || poe.getFiletype() == 'flv' || poe.getFiletype() == 'mov' || poe.getFiletype() == 'mp4' ||
poe.getFiletype() == 'mpeg' || poe.getFiletype() == 'avi' || poe.getFiletype() == 'mkv'}">
                                                media
                                            </c:when>
                                            <c:otherwise>
                                                document
                                            </c:otherwise>
                                        </c:choose>
                                        ">
                                        <div class="icon-space"></div>
                                        <div class="download-file"><a href="/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out
```

```
value='${poe.getFilename()}'></c:out>?download=true">Download</a></div>
                                    </div>
                                </c:otherwise>
                            </c:choose>

                            <span class="filename"><c:out value='${poe.getFilename()}'></c:out></span>
                            <c:if test="${partnerAndLatest}">
                                <form action="/api/deleteFile" method="post" class="delete-files">
                                    <input type="hidden" name="fileId" value="<c:out value='${poe.getId()}'></c:out>">
                                    <input type="hidden" name="deleteFile" value="<c:out value='${poe.getF_date() >
stages.get(stageIndex - 1).getDate() ? "true" : "false"}'></c:out>">
                                    <input type="hidden" name="projectId" value="<c:out value='${project.getId()}'></c:out>">
                                    <input type="hidden" name="fileName" value="<c:out value='${poe.getFilename()}'></c:out>">
                                    <input type="submit" value="" class="delete-icon">
                                </form>
                            </c:if>

                        </div>
                </c:if></c:if>
            </c:forEach>
            <c:if test="${partnerAndLatest}">
                <div class="new-image">
                    <form action="/uploadFile" method="post" enctype="multipart/form-data">
                        <input type="hidden" name="proj_id" value="<c:out value='${project.getId()}'></c:out>">
                        <input type="file" name="file">
                        <input type="hidden" name="stage" value="<c:out value='${stages.get(stageIndex).getId()}'></c:out>">
                        <input class="button" type="submit" name="submit" value="Upload">
                    </form>
                </div>
            </c:if>

            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">Review the claim and the attached proofs.</p>
            </c:if><c:if test="${partnerAndLatest}">
                <p class="status-message">Dell will now review your claim request.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Claim Rejected'}">
            <h3>Claim rejected</h3>
            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">Add a comment to explain your decision of rejecting the claim.</p>
            </c:if><c:if test="${partnerAndLatest}">
                <p class="status-message">Dell has rejected your claim. Look in the comments for an explanation.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Project Finished'}">
            <h3>Project finished!</h3>
            </div>
            <c:if test="${dellAndLatest}">
                <p class="status-message">Project is finished and now awaiting reimbursement.</p>
            </c:if><c:if test="${partnerAndLatest}">
                <p class="status-message">Claim has been approved and the project is now finished. Reimbursement is
processing.</p>
            </c:if>
        </c:if>

        <c:if test="${stages.get(stageIndex).getType() == 'Cancelled'}">
            <h3>Project has been cancelled</h3>
            </div>
            <p class="status-message">Project has been cancelled, look in the comments for further information.</p>
        </c:if>

        <c:if test="${dellAndLatest &&
                (project.getStatus() == 'Waiting Project Verification' || project.getStatus() == 'Waiting Claim Verification')}">
            <form method="post" class="stage-actions" action="/api/changeProjectStatus">
                <input type="hidden" name="currentType" value="${project.getStatus()}">
                <input type="hidden" name="projectId" value="${project.getId()}">
                <button name="answer" value="approved" class="green">Approve</button>
                <button name="answer" value="denied" class="red">Reject</button>
            </form>
        </c:if>
    </div>
</div>
```

Budgets.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

  <jsp:include page="header.jsp" />

  <div class="container" style="margin-top: 30px; padding-bottom: 30px;">
      <a href="/create-budget" class="button-clear u-pull-right">Create budget</a>
      <h3 style="margin-bottom: 1rem;">Budget view</h3>
      <p style="color: #9F9F9F;">Click on budget to edit it.</p>

      <div class="table-head">
          <span class="t-quarter">Quarter</span>
          <span class="t-year">Year</span>
          <span class="t-budget">Budget</span>
          <span class="t-reserved small">Reimbursed</span>
          <span class="t-reserved small">Reserved</span>
          <span class="t-reserved small">Available</span>
      </div>
      <c:forEach var="budget" items="${budgets}">
          <a href="/edit-budget?initialbudget=<c:out value='${budget.getInitial_budget()}' />&year=<c:out
value='${budget.getYear()}' />&quarter=<c:out value='${budget.getQuarter()}' />">
              <div class="project-item">
                  <span class="t-quarter"><c:out value="${budget.getQuarter()}" /></span>
                  <span class="t-year"><c:out value="${budget.getYear()}" /></span>
                  <span class="t-budget"><c:out value="${budget.getInitial_budget()}" />&#8364</span>
                  <span class="t-reserved small"><c:out value="${budget.getReimbursed()}" />&#8364</span>
                  <span class="t-reserved small"><c:out value="${budget.getReserved()}" />&#8364</span>
                  <span class="t-reserved small"><c:out value="${budget.getLeftAvailable()}" />&#8364</span>
              </div>
          </a>
      </c:forEach>
  </div>

  </body>
</html>
```

Create-budget.jsp

```
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="js/createcompany-functions.js"></script>
```

```
<div class="container">
<div class="big-paper">

    <h2>Create budget</h2>
    <form action="/createBudget" method="post">
        <div class="input-group">
            <span>Define budget</span>
            <input type="number" name="initial_budget" min="0" required>
        </div>
        <div class="input-group">
            <span>Select year</span>
            <select name="year">
            <c:forEach begin="2015" end="2020" varStatus="loop">
                <option value="${loop.index}">${loop.index}</option>
            </c:forEach>
            </select>
        </div>
        <div class="input-group">
            <span>Choose quarter</span>
            <select name="quarter">
                <c:forEach begin="1" end="4" varStatus="loop">
                    <option value="${loop.index}">${loop.index} </option>
                </c:forEach>
            </select>
        </div>

        <input class="button" type="submit" value="Set budget">
    </form>

  </div>
</div>




</body>
</html>
```

Create-company.jsp

```
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="js/createcompany-functions.js"></script>

<div class="container">
<div class="big-paper">
  <h2>Register new partner</h2>
  <form action="/api/createCompany" method="post">
    <div class="input-group">
        <span>Name of the partner</span>
        <input type="text" id="companyName" name="companyName">
    </div>
    <div class="input-group">
        <span>Select country</span>
        <select id="countryCode" name="countryCode">
            <option value="DK">Denmark</option>
            <option value="SE">Sweden</option>
            <option value="NO">Norway</option>
            <option value="DE">Germany</option>
            <option value="PL">Poland</option>
            <option value="CZ">Czech</option>
        </select>
    </div>

    <div class="logo-box" style="float: left; clear: left;">
        <span>Select logo</span>
        <div id="logo-results"></div>
        <div id="branding"  style="float: left;"></div><br>
        <span>Or upload your own</span>
        <input type="file" name="logo">
        <input type="hidden" name="logoUrl" id="logo-url">
    </div>
    <input class="button" type="submit" value="Create Company">
  </form>

  </div>
</div>

</body>
</html>
```

Create-user.jsp

```
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="js/createcompany-functions.js"></script>

<div class="container">
<div class="big-paper">

    <h2> Register new user </h2>
    <form action="/createUser" method="post">
        <div class="input-group">
            <span>Select partner</span>
            <select name="selectedCompany">
                <c:forEach var="companies" items="${companies}">
                    <option value="<c:out value='${companies.getId()}'></c:out>" <c:if
test="${companies.getName().equals(partnerName)}">selected</c:if>><c:out
value='${companies.getName()}'></c:out></option>
                </c:forEach>
            </select>
        </div>
        <div class="input-group">
            <span>Full name</span>
            <input type="text" name="userName">
        </div>
        <div class="input-group">
            <span>Email</span>
            <input type="email" name="userEmail">
        </div>

        <p style="float:left; clear: left;">Users will set their password via email.</p>

        <input class="button" type="submit" value="Register User">
    </form>

  </div>
```

```
</div>




</body>
</html>
```

Create-project.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />

<div class="container project" style="margin-top: 60px;">
  <H1>Project request</H1>

  <form action="/project-request" method="post">
    <span><c:out value="${User.getCompany().getName()}" /> is requesting</span>
    <div class="amount-box">
      <span class="small-label">Euro is binding currency</span>
      <input class="amount" name="budget" type="text" placeholder="Amount"/>
      <span class="euro-label">&#8364</span>
    </div>
    <span>for a </span>
      <input class="amount custom-type" type="text" name="type" id="type">

    <span style="clear: left;">With execution scheduled</span>
    <select name="execution_year" id="year">
      <option value="2015">2015</option>
      <option value="2016">2016</option>
    </select>
    <select name="execution_month" id="month">
      <option value="1">January</option>
      <option value="2">February</option>
      <option value="3">March</option>
      <option value="4">April</option>
      <option value="5">May</option>
      <option value="6">June</option>
      <option value="7">July</option>
      <option value="8">August</option>
      <option value="9">September</option>
      <option value="10">October</option>
      <option value="11">November</option>
      <option value="12">December</option>
    </select>
    <span class="add_day">Add day</span>
    <select name="execution_day" id="day"style="display: none">
      <option value='0'>0</option>
    </select>
    <textarea name="body" id="description" placeholder="Describe your project here."></textarea>
    <button type="submit" class="button">Send request</button>
  </form>
</div>



<script>
  $('span.add_day').click(function() {
    setDays();
    $('span.add_day').css('display', 'none');
    $('select#day').css('display', 'block');
    $('select#day').addClass('visible');
  });

  $('select#month option').each(function(){
      var d = new Date();
      var m = d.getMonth();

      if($(this).val() < m+1) {
          $(this).remove();
      }
  });

  var searchNext = true;

  var removeNotMatches = function(list, q, sync) {
          var matches = [];
          var regex = new RegExp(q, "i");
          for (var i = 0; i < list.length; i++) {
              if (regex.test(list[i])) {
                  if (list == pres) {
                      matches.push(list[i])
                  } else if (pres.indexOf(list[i]) < 0) {
                      matches.push(list[i])
          }}}
          return matches;
  }


  var pres = ["Online advertising",
      "Billboard ad",
      "TV Promotion",
      "Face-to-face event",
        "Webinar",
        "Direct mail"];

  var preset = function(q, sync) {
      if(q == '')
          sync(pres)
      else
          sync(removeNotMatches(pres, q));
  }

  var types = [];

  var typesFilter = function(q, sync, async) {
      if(q.length == 3 && searchNext) {
          return $.ajax({
              dataType: "json",
              url: "/getTypes",
              data: {query: q},
              success: function(data) {
                  types = data;
                  async(removeNotMatches(types));
              }
          });
      } else
          return removeNotMatches(types, q, sync);
  }

  $('#type').typeahead({
              hint: false,
              highlight: true,
              minLength: 0
          }, {
              name: 'preset',
              source: preset
          }, {
              name: "type",
              source: typesFilter
          }

  ).change(function() {
              if($(this).val().length < 3)
                  searchNext = true;
              if($(this).val().length > 4)
                  searchNext = false;
          });

  $('select#month').change(function() {
    if($('select#day').hasClass('visible')){
      setDays();
    }
  });

  function setDays() {
    var days = 31;
    var e = document.getElementById("month");
    var strMonth = e.options[e.selectedIndex].value;

    if(strMonth == "1" || strMonth == "3" || strMonth == "5" ||
              strMonth == "7" || strMonth == "8" || strMonth == "10" || strMonth == "12") {
      days = 31;
    } else if(strMonth == "4" || strMonth == "6" || strMonth == "9" || strMonth == "11") {
      days = 30;
    } else if(strMonth == "2") {
      days = 28;
    }

    $('select#day').html("");
    for(var i=1; i<=days; i++) {
      $('select#day').append("<option value='" + i + "'>" + i +".</option>");
    }
  }
</script>

</body>
</html>
```

Edit-budget.jsp

```
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="js/createcompany-functions.js"></script>

<div class="container">
    <div class="big-paper">
        <h2>Edit budget</h2>
        <form action="/modifyBudget" method="post">
            <div class="input-group">
                <span>Define budget</span>
                <input type="number" name="newBudget" min="0" required value="<c:out value='${initialbudget}' />">
            </div>
            <div class="input-group">
                <span>Select year</span>
                <select name="year">
                <c:forEach begin="2015" end="2020" varStatus="loop">
                    <option value="${loop.index}" <c:if test="${loop.index.equals(year)}">selected</c:if>
>${loop.index}</option>
                </c:forEach>
                </select>
            </div>
            <div class="input-group">
                <span>Choose quarter</span>
                <select name="quarter">
                    <c:forEach begin="1" end="4" varStatus="loop">
                        <option value="${loop.index}" <c:if test="${loop.index.equals(quarter)}">selected</c:if> >${loop.index}
</option>
                    </c:forEach>
                </select>
            </div>

            <input class="button" type="submit" value="Edit budget">
        </form>

    </div>
</div>

</body>
</html>
```

Header.jsp

```
<%@ page import="Domain.User" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Dell</title>
  <meta name="description" content="Dell campaign management system">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="apple-touch-icon" sizes="57x57" href="favicon/apple-touch-icon-57x57.png">
    <link rel="apple-touch-icon" sizes="60x60" href="favicon/apple-touch-icon-60x60.png">
    <link rel="apple-touch-icon" sizes="72x72" href="favicon/apple-touch-icon-72x72.png">
    <link rel="apple-touch-icon" sizes="76x76" href="favicon/apple-touch-icon-76x76.png">
    <link rel="apple-touch-icon" sizes="114x114" href="favicon/apple-touch-icon-114x114.png">
    <link rel="apple-touch-icon" sizes="120x120" href="favicon/apple-touch-icon-120x120.png">
    <link rel="apple-touch-icon" sizes="144x144" href="favicon/apple-touch-icon-144x144.png">
    <link rel="apple-touch-icon" sizes="152x152" href="favicon/apple-touch-icon-152x152.png">
    <link rel="apple-touch-icon" sizes="180x180" href="favicon/apple-touch-icon-180x180.png">
    <link rel="icon" type="image/png" href="favicon/favicon-32x32.png" sizes="32x32">
    <link rel="icon" type="image/png" href="favicon/favicon-194x194.png" sizes="194x194">
    <link rel="icon" type="image/png" href="favicon/favicon-96x96.png" sizes="96x96">
    <link rel="icon" type="image/png" href="favicon/android-chrome-192x192.png" sizes="192x192">
    <link rel="icon" type="image/png" href="favicon/favicon-16x16.png" sizes="16x16">
    <link rel="manifest" href="favicon/manifest.json">
    <meta name="msapplication-TileColor" content="#2d89ef">
    <meta name="msapplication-TileImage" content="favicon/mstile-144x144.png">
    <meta name="theme-color" content="#00a3ff">

  <link href='http://fonts.googleapis.com/css?family=Roboto:400,300,500&subset=latin,latin-ext' rel='stylesheet'
type='text/css'>
  <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
  <link rel="stylesheet" href="css/jquery.fancybox.css" type="text/css" media="screen" />
  <script type="text/javascript" src="js/jquery.fancybox.pack.js"></script>
  <script type="text/javascript" src="js/moment.js"></script>
```

```
<script type="text/javascript" src="js/typeahead.bundle.min.js"></script>
<script type="text/javascript" src="js/jensabox.js"></script>
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<link href="css/normalize.css" rel="stylesheet" media="all">
<link href="css/skeleton.css" rel="stylesheet" media="all">
<link href="css/style.css" rel="stylesheet" media="all">

<script>
 function formatDates() {
   $('span.isDate').each(function() {
     var milis = parseInt($(this).text());
     var time = moment(milis).format('Do MMMM YYYY, H:mm');
     $(this).text(time);

     $(this).removeClass('isDate');
   });
     $('.isShortDate').each(function() {
         var millis = parseInt($(this).text());
         var m = millis.toString()
         if(millis == 0)
             $(this).text("N/A");
         else if(m.substring(m.length - 4, m.length) == 1000)
             $(this).text(moment(millis).format('MMM YYYY'));
         else
             $(this).text(moment(millis).format('MMM D[.] YYYY'));
     })
 }

 function highlightInputs() {
     if($('.notification.error').length > 0) {
         var $this = $('.notification.error');
         var inputErrorField = $this.text().split("|")[1].split(",");
         console.log("errorInputFields: " + inputErrorField);
         $this.text($this.text().substring(0, $this.text().indexOf("|")));

         for(var i = 0; i < inputErrorField.length; i++) {
             $("input[name=" + inputErrorField[i] + "]").addClass("error");
             $("textarea[name=" + inputErrorField[i] + "]").addClass("error");
             $("select[name=" + inputErrorField[i] + "]").addClass("error");
         }
     }
 }


 $(document).ready(function() {
   formatDates();
   highlightInputs();

     <c:if test="${formData != null}">
     var formData = [
         <c:forEach items="${formData}" var="d">
         ['<c:out value="${d[0]}" />','<c:out value="${d[1]}" escapeXml="false" />'],
         </c:forEach>
     ];
     for(var i = 0; i <formData.length; i++) {
         $('input[name=' + formData[i][0] + ']').val(formData[i][1]);
         $('textarea[name=' + formData[i][0] + ']').val(formData[i][1]);
         $('select[name=' + formData[i][0] + ']').val(formData[i][1]);
     }
     </c:if>

     var toggles = document.querySelectorAll(".cmn-toggle-switch");

     for (var i = toggles.length - 1; i >= 0; i--) {
         var toggle = toggles[i];
         toggleHandler(toggle);
     };
```

```
    function toggleHandler(toggle) {
      toggle.addEventListener( "click", function(e) {
        e.preventDefault();
        (this.classList.contains("active") === true) ? this.classList.remove("active") : this.classList.add("active");
        var menu = $('div.mobile-menu');
        if(menu.hasClass('active')) {
          menu.removeClass('active');
          setTimeout(function(){
            menu.css('visibility','hidden');
          },500);
        } else {
          menu.css('visibility','visible');
          menu.addClass('active');
        }
      });
    }
  });
 </script>
</head>
<body onload="document.body.setAttribute('class','loaded')">
<c:set var="uri" value="${pageContext.request.requestURI}" />
<c:if test="${errorMessage != null}">
  <div class="notification error"><c:out value="${sessionScope.errorMessage}"></c:out></div>
  <c:if test="${sessionScope.deleteE != null}">
    <c:remove var="formData" scope="session" />
    <c:remove var="errorMessage" scope="session" />
  </c:if>
  <c:if test="${sessionScope.errorMessage != null}">
    <c:set var="deleteE" scope="session" value="true"></c:set>
  </c:if>
</c:if><c:if test="${message != null}">
  <div class="notification message"><c:out value="${sessionScope.message}"></c:out></div>
  <c:if test="${sessionScope.deleteM != null}">
    <c:remove var="message" scope="session" />
  </c:if>
  <c:if test="${sessionScope.message != null}">
    <c:set var="deleteM" scope="session" value="true"></c:set>
  </c:if>
</c:if>

<div class="header u-full-width">
 <div class="container">
  <a href="/dashboard">
   <div class="logo u-pull-left">
    <img src="img/small_dell_logo.svg" alt="Dell logo">
    <span>Campaign<br/>management<br/>system</span>
   </div>
  </a>
  <div class="user-label u-pull-right">
   <img src="img/white_dropdown.svg" alt="Logout menu">
   <span><c:out value="${User.getName()}"></c:out></span>
   <ul class="submenu">
    <li><a href="/logout">Logout</a></li>
   </ul>
  </div>
  <c:if test="${User.getCompany_id() == 1}">
    <div class="mobile u-pull-right" style="margin-top: 23px;">
      <button class="cmn-toggle-switch cmn-toggle-switch__htx">
        <span>toggle menu</span>
      </button>
    </div>
    <div class="u-pull-right desktop">
     <a href="/statistics" class="head-button <c:if test="${fn:contains(uri, 'statistics')}">active</c:if>">Stats</a>
     <a href="/budgets" class="head-button <c:if test="${fn:contains(uri, 'budget')}">active</c:if>">Budgets</a>
     <a href="/users" class="head-button <c:if test="${fn:contains(uri, 'user')}">active</c:if>">Users</a>
     <a href="/partners" class="head-button <c:if test="${fn:contains(uri, 'partner')}">active</c:if>">Partners</a>
```

```
          <a href="/dashboard" class="head-button <c:if test="${fn:contains(uri, 'index')}">active</c:if>">Dashboard</a>
        </div>
        <div class="budget-label u-pull-left">

          <c:if test="${activeBudget != null}">
            <span class="big"><c:out value="${activeBudget.getLeftAvailable()}"></c:out>&#8364 <strong>(<c:out
value="${activeBudget.getReserved()}"></c:out>&#8364 reserved)</strong></span>
            <span class="desc">is left available in this quarter</span>
          </c:if><c:if test="${activeBudget == null}">
            <span class="big"> <a href="/budgets">Set budget</a> </span>
            <span class="desc">No budget available</span>
          </c:if>

        </div>
    </c:if>
      <c:if test="${User.getCompany_id() != 1}">
        <div class="u-pull-right">
          <a href="/project-request" class="project-request button">Project request</a>
        </div>
      </c:if>

  </div>
</div>

<div class="mobile-menu" id="mobile-menu">
    <a href="/budgets" class="head-button <c:if test="${fn:contains(uri, 'budget')}">active</c:if>">Budgets</a>
    <a href="/users" class="head-button <c:if test="${fn:contains(uri, 'user')}">active</c:if>">Users</a>
    <a href="/partners" class="head-button <c:if test="${fn:contains(uri, 'partner')}">active</c:if>">Partners</a>
    <a href="/dashboard" class="head-button <c:if test="${fn:contains(uri, 'index')}">active</c:if>">Dashboard</a>
</div>
```

Index.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<jsp:include page="header.jsp" />

  <div class="container actions" style="margin-top: 40px;">
      <c:if test="${User.getCompany_id() == 1}">
      <a href="/dashboard">
        <div class="filter <c:if test="${param.state == null}">active</c:if>">
          <div class="circle waiting"><c:out value="${statusCount[0]}" /> </div>
          <span>Waiting for action</span>
        </div>
      </a>
      <a href="?state=inExecution">
        <div class="filter <c:if test="${param.state != null && param.state.equals('inExecution')}">active</c:if>">
          <div class="circle execution"><c:out value="${statusCount[1]}" /></div>
          <span>In execution</span>
        </div>
      </a>
      </c:if>
      <c:if test="${User.getCompany_id() != 1}">
       <a href="/dashboard">
        <div class="filter <c:if test="${param.state == null}">active</c:if>">
          <div class="circle execution"><c:out value="${statusCount[0]}" /></div>
         <span>Active</span>
        </div>
       </a>
```

```
   </c:if>
  <a href="?state=finished">
   <div class="filter <c:if test="${param.state != null && param.state.equals('finished')}">active</c:if>">
    <div class="circle finished"><c:out value="${statusCount[2]}" /></div>
    <span>Finished</span>
   </div>
  </a>
  <div class="searchbox">
    <input type="text" placeholder="Search" class="search"
        <c:if test="${param.type != null}">value="<c:out value="${param.type}"></c:out>"</c:if>
        <c:if test="${param.company != null}">value="<c:out value="${param.company}"></c:out>"</c:if>
        <c:if test="${param.state != null}">value="<c:out value="${param.state}"></c:out>"</c:if>
        <c:if test="${param.q != null}">value="<c:out value="${param.q}"></c:out>"</c:if>
        />
  </div>

 </div>

 <div class="container" style="margin-top: 30px; padding-bottom: 30px;">

   <c:if test="${projects != null}">
     <div class="table-head">
       <span class="id">ID</span>
       <span class="partner">Partner</span>
       <span class="type">Type</span>
       <span class="state">State</span>
       <span class="execution-date">Execution date</span>
     </div>

   <c:forEach var="project" items="${projects}">

     <a href="/project?id=<c:out value="${project.getId()}" />">
       <div class="project-item
       <c:if test="${User.getCompany_id() == 1}">
        <c:if test="${project.isUnread_admin()}">unread</c:if>
       </c:if>
       <c:if test="${User.getCompany_id() != 1}">
        <c:if test="${project.isUnread_partner()}">unread</c:if>
       </c:if>
       ">
         <span class="id"><strong>#</strong><c:out value="${project.getId()}" /></span>
         <span class="partner"><c:out value="${project.getCompanyName()}" /></span>
         <span class="type"><c:out value="${project.getType()}" /></span>
         <c:choose>
           <c:when test="${project.getNotification() != null && ((User.getCompany_id() == 1 &&
project.isUnread_admin()) || (User.getCompany_id() != 1 && project.isUnread_partner()))}">
             <span class="notification"><c:out value="${project.getNotification()}" /></span>
           </c:when>
           <c:otherwise>
             <span class="state small"><c:out value="${project.getStatus()}" /></span>
           </c:otherwise>
         </c:choose>
         <span class="execution-date small isShortDate"><c:out value="${project.getF_execution_date()}"></c:out>
</span>
       </div>
     </a>
   </c:forEach>
   </c:if><c:if test="${results != null && projects == null}">


  <c:forEach items="${results}" var="container">
    <h5><c:out value="${container.getType()}"></c:out>s</h5>
    <div class="table-head">
      <span class="id">ID</span>
      <span class="type">Type</span>
      <span class="result">Result</span>
    </div>
    <c:forEach items="${container.getContainer()}" var="result">

      <a href="<c:if test="${result.getType() == 'User'}">/user?id=</c:if><c:if test="${result.getType() !=
'User'}">/project?id=</c:if><c:out value="${result.getId()}"></c:out>">
        <div class="project-item">
          <span class="id"><strong>#</strong><c:out value="${result.getId()}"></c:out></span>
          <span class="type"><c:out value="${result.getType()}"></c:out></span>
          <span class="result"><c:out value="${fn:substring(result.getBody(), 0, 70)}" ></c:out></span>
        </div>
      </a>
    </c:forEach>
  </c:forEach>
 </c:if>
 </div>

<script type="application/javascript">
  $(document).ready(function() {
    var searchNext = true;

    var removeNotMatches = function(list, q) {
      var matches = [];

      var regex = new RegExp(q, "i");

      for(var i = 0; i < list.length; i++) {
        if(regex.test(list[i]))
          matches.push(list[i])
      }

      return matches;
    }

    var statuses = new Bloodhound({
      datumTokenizer: Bloodhound.tokenizers.whitespace,
      queryTokenizer: Bloodhound.tokenizers.whitespace,
      local: ["Waiting Project Verification",
        "Project Rejected",
        "Project Approved",
        "Waiting Claim Verification",
        "Claim Rejected",
        "Project Finished",
        "Cancelled"]
    });

    var types = [];

    var typesFilter = function(q, sync, async) {
      if(q.length == 3 && searchNext) {
        return $.ajax({
          dataType: "json",
          url: "/getTypes",
          data: {query: q},
          success: function(data) {
            types = data;
            console.log(types);
            async(types);
          }
        });
      } else
        sync(removeNotMatches(types, q));
    }


    var companies = [];
    var companyFilter = function(q, sync, async) {
      if(<c:out value="${User.getCompany_id()}"></c:out> === 1) {
        if (q.length == 3 && searchNext) {
                return $.ajax({
                    dataType: "json",
                    url: "/getCompanyNames",
                    data: {query: q},
                    success: function (data) {
                        companies = data;
                        async(companies);
                        return companies;
                    }
                });
            } else
                sync(removeNotMatches(companies, q));
        }
    }

    var lsFilter = function(q, sync) {
        if(localStorage.typeahead != null)
            sync(removeNotMatches(localStorage.typeahead.split(","), q));
    }

    $('.search').typeahead({
        hint: false,
        highlight: true,
        minLength: 1
            }, {
        name: "typeahead",
        source: lsFilter,
        limit: 3,
                templates: {
                    header: '<h3 class="typeahead-header">History</h3>'
                }
    }, {
        name: 'statuses',
        source: statuses,
        limit: 4,
        templates: {
            header: '<h3 class="typeahead-header">Statuses</h3>'
        }
        }, {
        name: "type",
        source: typesFilter,
        limit: 4,
        templates: {
            header: '<h3 class="typeahead-header">Types</h3>'
        }
        }, {
        name: "companies",
        source: companyFilter,
        limit:4,
        templates: {
            header: '<h3 class="typeahead-header">Companies</h3>'
        }
    }

    ).bind("typeahead:selected", function () {
                console.log("ed");
                onTypeaheadSelect();
    }).keypress(function(e) {
                if(e.keyCode == 13) { // enter
                    var val = $(".search").val();
                    if(val.length > 3 && (localStorage.typeahead == null ||
localStorage.typeahead.toLowerCase().indexOf(val.toLowerCase()) == -1))
                        localStorage.typeahead = localStorage.typeahead == undefined ? val : localStorage.typeahead + "," + val;
                    if(val.substr(0, 1) == "#" && !isNaN(val.substr(1)))
                        window.location.href = "/project?id=" + val.substr(1);
                    else
                        window.location.href = "/dashboard?q=" + val;
                }
       }).change(function() {
                if($(this).val().length < 3)
                searchNext = true;
                if($(this).val().length > 4)
                searchNext = false;
              });
      var onTypeaheadSelect = function() {
         var header, selected = $(".search").val();
         $('.tt-suggestion').each(function() {
            if($(this).text() == selected)
               header = $(this).parent().find("h3").text()
         })
         if(header == "Statuses"){
            window.location.href = "/dashboard?state=" + selected;
         }else if(header == "Types") {
            window.location.href = "/dashboard?type=" + selected;
         }else if(header == "Companies")
            window.location.href = "/dashboard?company=" + selected;
      }


      $('span.twitter-typeahead').css('width', '100%');
      $('div.searchbox input').css('position', 'absolute !important');
   });
</script>

  </body>
</html>
```

Login.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <meta charset="utf-8">
    <title>Dell</title>
    <meta name="description" content="Dell campaign management system">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href='http://fonts.googleapis.com/css?family=Roboto:400,300,500&subset=latin,latin-ext' rel='stylesheet'
type='text/css'>
    <link href="/css/normalize.css" rel="stylesheet" media="all">
    <link href="/css/skeleton.css" rel="stylesheet" media="all">
    <link href="/css/style.css" rel="stylesheet" media="all">
</head>
<body>
  <div class="login-background u-pull-left">

    <div class="container login-container">
      <div class="row">
        <div class="eight columns" style="padding-right: 30px;">
          <h1>Login instructions</h1>
          <p>Please fill-in login credentials. Testing login credentials for DELL access are: <br>
          Email: honason@gmail.com <br>
          Password: horse <br/><br/>
          Credentials for PARTNER access are: <br/>
          Email: Anden702@gmail.com <br>
          Password: hest
          </p>
        </div>
        <div class="four columns login-form">
          <div class="message u-full-width u-pull-left">
              <% if (request.getAttribute("message") != null) { %>
                <p><%= request.getAttribute("message") %></p>
              <% };%>
          </div>
          <form action="/login" method="post" class="u-full-width" autocomplete="off">
            <input type="hidden" name="action" value="login">
            <input class="u-full-width just-line" type="text" name="email" placeholder="email">
            <input class="u-full-width just-line" type="password" name="password" placeholder="password">
            <input class="u-full-width button submit" type="submit" value="login">
          </form>
        </div>
      </div>
    </div>

  </div>

</body>
</html>
```

Partner.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

 <jsp:include page="header.jsp" />

 <div class="container" style="margin-top: 30px; padding-bottom: 30px;">
    <c:if test="${partner.getImg_filename() != null}" >
        <img class="company-image" src="/resources/companies/<c:out value='${partner.getId()}' />/<c:out
value='${partner.getImg_filename()}' />" />
    </c:if>
    <h3 class="company-name"><c:out value='${partner.getName()}' /></h3>
    <p class="company-desc">You can see all <c:out value='${partner.getName()}' />'s projects below.</p>

    <a href="/create-user?partnerName=<c:out value='${partner.getName()}' />" class="button-clear u-pull-right">Add
user</a>

    <h5 class="above-list">Users</h5>
    <div class="table-head">
        <span class="id">ID</span>
        <span class="user-name">Name</span>
        <span class="email">Email</span>
    </div>
    <c:forEach var="user" items="${users}">
        <a href="user?id=<c:out value="${user.getId()}" />">
            <div class="project-item">
                <span class="id"><strong>#</strong><c:out value="${user.getId()}" /></span>
                <span class="user-name"><c:out value="${user.getName()}" /></span>
                <span class="email"><c:out value="${user.getEmail()}" /></span>
            </div>
        </a>
    </c:forEach>

    <h5 class="above-list">Projects</h5>
    <div class="table-head">
        <span class="id">ID</span>
        <span class="partner">Partner</span>
        <span class="type">Type</span>
        <span class="state">State</span>
        <span class="execution-date">Execution date</span>
    </div>
    <c:forEach var="project" items="${projects}">
        <a href="/project?id=<c:out value="${project.getId()}" />">
            <div class="project-item">
```

```
                <span class="id"><strong>#</strong><c:out value="${project.getId()}" /></span>
                <span class="partner"><c:out value='${partner.getName()}' /></span>
                <span class="type"><c:out value="${project.getType()}" /></span>
                <span class="state small"><c:out value="${project.getStatus()}" /></span>
                <span class="execution-date small isShortDate"><c:out
value="${project.getExecution_date()}"></c:out></span>
            </div>
        </a>
    </c:forEach>
  </div>

  </body>
</html>
```

Partners.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

  <jsp:include page="header.jsp" />

  <div class="container" style="margin-top: 30px; padding-bottom: 30px;">
    <a href="/create-company" class="button-clear u-pull-right">Add partner</a>
    <h3 style="margin-bottom: 1rem;">Partners</h3>
    <p style="color: #9F9F9F;">Click on partner to get to partner's page.</p>

    <div class="table-head">
        <span class="id">ID</span>
        <span class="partner-name">Company name</span>
        <span class="country">Country</span>
    </div>

    <c:forEach var="partner" items="${partners}">
        <a href="partner?id=<c:out value='${partner.getId()}' />" />
        <div class="project-item">
            <span class="id"><strong>#</strong><c:out value="${partner.getId()}" /></span>
            <span class="partner-name"><c:out value="${partner.getName()}" /></span>
            <span class="country small"><c:out value="${partner.country_code()}" /></span>
        </div>
        </a>
    </c:forEach>
  </div>

  </body>
</html>
```

Project.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ include file="header.jsp" %>

<c:set var="stageIndex" value="0"></c:set>
<c:set var="messageIndex" value="0"></c:set>
```

```
<c:set var="lastStage" value="false"></c:set>

<c:if test="${project.getMessage() != null}"><c:out value="${project.getMessage()}"></c:out></c:if>

<div class="container project-container">
    <div class="u-pull-right" style="margin-bottom: 20px;">
        <c:if test="${(project.getStatus() != 'Project Finished')}">
            <c:if test="${(project.getStatus() != 'Cancelled')}">
                <div class="cancel-box"><a href="#" class="cancel-button jensabox-trigger">Cancel Project</a></div>
            </c:if>
        </c:if>
        <div class="project-state"><c:out value="${project.getStatus()}" /></div>
        <span class="state">State</span>
    </div>
    <h1><span style="color: #9F9F9F;">#<c:out value="${project.getId()}" /></span> <c:out value="${project.getType()}"
/></h1>

    <div class="project-items">
    <c:if test="${(stages.size() + messages.size()) > 0}">
        <c:forEach var="i" begin="0" end="${stages.size() + messages.size() - 1}">
            <c:if test="${stages.size() - 1 == stageIndex}">
                <c:set var="lastStage" value="true"></c:set>
            </c:if>
            <c:choose>
                <c:when test="${stageIndex == -1}">
                    <%@ include file="shortcodes/messageItemByIndex.jsp" %>
                    <c:set var="messageIndex" value="${messageIndex + 1}"></c:set>
                </c:when>
                <c:when test="${messageIndex == -1}">
                    <%@ include file="shortcodes/stageItemByIndex.jsp" %>
                    <c:set var="stageIndex" value="${stageIndex + 1}"></c:set>
                </c:when>
                <c:otherwise>
                    <c:if test="${stages.size() > 0 && messages.size() > 0}">
                    <c:choose>
                        <c:when test="${stages.get(stageIndex).getDate() <
messages.get(messageIndex).getCreation_date_millis()}">
                            <%@ include file="shortcodes/stageItemByIndex.jsp" %>
                            <c:set var="stageIndex" value="${stageIndex + 1}"></c:set>
                            <c:if test="${stageIndex == stages.size()}">
                                <c:set var="stageIndex" value="-1"></c:set>
                            </c:if>
                        </c:when>
                        <c:otherwise>
                            <%@ include file="shortcodes/messageItemByIndex.jsp" %>
                            <c:set var="messageIndex" value="${messageIndex + 1}"></c:set>
                            <c:if test="${messageIndex == messages.size()}">
                                <c:set var="messageIndex" value="-1"></c:set>
                            </c:if>
                        </c:otherwise>
                    </c:choose>
                    </c:if>
                    <c:if test="${messages == null || messages.size() == 0}">
                        <%@ include file="shortcodes/stageItemByIndex.jsp" %>
                        <c:set var="stageIndex" value="${stageIndex + 1}"></c:set>
                    </c:if><c:if test="${stages == null || stages.size() == 0}">
                        <%@ include file="shortcodes/messageItemByIndex.jsp" %>
                        <c:set var="messageIndex" value="${messageIndex + 1}"></c:set>
                    </c:if>
                </c:otherwise>
            </c:choose>
        </c:forEach>
    </c:if>
        <c:set var="stageIndex" value="${stages.size() - 1}"></c:set>

    <c:if test="${User.getCompany_id() != 1 && (project.getStatus() == 'Project Rejected' || project.getStatus() == 'Project
Approved' || project.getStatus() == 'Claim Rejected')}">
```

```
        <div class="item pull-right u-full-width">
            <div class="bubble">

                <c:if test="${project.getStatus() == 'Project Approved' || project.getStatus() == 'Claim Rejected'}">
                    <div class="inner-bubble">
                        <h3>Proof Of Execution</h3>
                        <p class="instructions">Upload your images and documents, one by one.</p>
                        <c:forEach items="${poes}" var="poe" varStatus="ite" >
                            <c:if test="${poe.getF_deletion_date() == 0}">
                                <div class="proof-container <c:if test="${poe.getF_date() > stages.get(stageIndex - 1).getDate()}">
new</c:if>">
                                    <c:choose>
                                        <c:when test="${poe.getFiletype() == 'jpg' || poe.getFiletype() == 'png' || poe.getFiletype() == 'jpeg'
|| poe.getFiletype() == 'gif' || poe.getFiletype() == 'bmp'}">
                                            <div class="proof" style="background-image: url(/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out value='${poe.getFilename()}'></c:out>)">
                                                <a class="fancybox" rel="<c:out value='${poe.getProj_id()}'></c:out>"
href="/resources/<c:out value='${poe.getProj_id()}'></c:out>/<c:out value='${poe.getFilename()}'></c:out>"><div
class="view-image"></div></a>
                                                <div class="download-file"><a href="/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out
value='${poe.getFilename()}'></c:out>?download=true">Download</a></div>
                                            </div>
                                        </c:when>
                                        <c:otherwise>
                                            <div class="proof
                                            <c:choose>
                                                <c:when test="${poe.getFiletype() == 'xlsx' || poe.getFiletype() == 'xls' || poe.getFiletype() ==
'numbers' || poe.getFiletype() == 'xml'}">
                                                    excel
                                                </c:when>
                                                <c:when test="${poe.getFiletype() == 'zip' || poe.getFiletype() == 'rar' || poe.getFiletype() ==
'tar' || poe.getFiletype() == 'dmg'}">
                                                    archive
                                                </c:when>
                                                <c:when test="${poe.getFiletype() == 'mp3' || poe.getFiletype() == 'flac' || poe.getFiletype()
== 'm4a' || poe.getFiletype() == 'wav' || poe.getFiletype() == 'flv' || poe.getFiletype() == 'mov' || poe.getFiletype() == 'mp4' ||
poe.getFiletype() == 'mpeg' || poe.getFiletype() == 'avi' || poe.getFiletype() == 'mkv'}">
                                                    media
                                                </c:when>
                                                <c:otherwise>
                                                    document
                                                </c:otherwise>
                                            </c:choose>
                                            ">
                                                <div class="icon-space"></div>
                                                <div class="download-file"><a href="/resources/<c:out
value='${poe.getProj_id()}'></c:out>/<c:out
value='${poe.getFilename()}'></c:out>?download=true">Download</a></div>
                                            </div>
                                        </c:otherwise>
                                    </c:choose>

                                    <span class="filename"><c:out value='${poe.getFilename()}'></c:out></span>
                                    <form action="/api/deleteFile" method="post" class="delete-files">
                                        <input type="hidden" name="fileId" value="<c:out value='${poe.getId()}'></c:out>">
                                        <input type="hidden" name="deleteFile" value="<c:out value='${poe.getF_date() >
stages.get(stageIndex - 1).getDate() ? "true" : "false"}'></c:out>">
                                        <input type="hidden" name="projectId" value="<c:out value='${project.getId()}'></c:out>">
                                        <input type="hidden" name="fileName" value="<c:out value='${poe.getFilename()}'></c:out>">
                                        <input type="submit" value="" class="delete-icon">
                                    </form>
                                </div>
                            </c:if>
                        </c:forEach>
                        <div class="new-image">
                            <form action="/uploadFile" method="post" enctype="multipart/form-data">
```

```
                                <input type="hidden" name="proj_id" value="<c:out value='${project.getId()}'></c:out>">
                                <input type="file" name="file">
                                <input class="button" type="submit" name="submit" value="Upload">
                            </form>
                        </div>
                    </div>
                </c:if>

                <form method="post" action="/api/changeProjectStatus">
                    <input type="hidden" name="currentType" value="${project.getStatus()}">
                    <input type="hidden" name="projectId" value="${project.getId()}">

                    <c:if test="${project.getStatus() == 'Project Rejected'}">
                    <div class="inner-bubble project">
                        <h3>Resubmit Project</h3>
                        <span><c:out value="${User.getCompany().getName()}" /> is requesting</span>
                        <div class="amount-box">
                            <span class="small-label">Euro is binding currency</span>
                            <input class="amount" name="budget" type="text" placeholder="Amount"/>
                            <span class="euro-label">&#8364</span>
                        </div>
                        <span>for a</span>
                        <input class="amount custom-type" type="text" name="type" id="type">
                        <span style="clear: left;">With execution scheduled</span>
                        <select name="execution_year" id="year">
                            <option value="2015">2015</option>
                            <option value="2016">2016</option>
                        </select>
                        <select name="execution_month" id="month">
                            <option value="1">January</option>
                            <option value="2">February</option>
                            <option value="3">March</option>
                            <option value="4">April</option>
                            <option value="5">May</option>
                            <option value="6">June</option>
                            <option value="7">July</option>
                            <option value="8">August</option>
                            <option value="9">September</option>
                            <option value="10">October</option>
                            <option value="11">November</option>
                            <option value="12">December</option>
                        </select>
                        <span class="add_day">Add day</span>
                        <select name="execution_day" id="day"style="display: none">
                            <option value='0'>0</option>
                        </select>
                        <textarea name="body" id="description" placeholder="Describe your project here."></textarea>

                    </div>
                        <div class="stage-actions" style="margin-top: 20px;">
                            <button name="answer" value="approved" class="blue">Resubmit project</button>
                        </div>
                    </c:if><c:if test="${project.getStatus() == 'Project Approved'}">
                    <p class="status-message">When the project is finished, upload images and documents as a proof of
execution.</p>
                    <div class="stage-actions">
                        <button name="answer" value="approved" class="blue">Send</button>
                    </div>

                </c:if><c:if test="${project.getStatus() == 'Claim Rejected'}">
                    <p class="status-message">Resubmit claim, upload new proof of execution.</p>
                    <div class="stage-actions">
                        <button name="answer" value="approved" class="blue">Resubmit claim</button>
                    </div>
                </c:if>
                </form>
            </div>
```

```
        </div>
    </c:if>
    </div>


    <form>
        <input type="hidden" name="userId" id="userId" value="${User.getId()}" />
        <input type="hidden" name="projectId" id="projectId" value="${project.getId()}" />
        <input type="hidden" name="companyId" id="companyId" value="${User.getCompany_id()}" />
        <textarea name="body" id="message" placeholder="Write your message"></textarea>
        <button id="submitMessage" class="submit">Send message</button>
    </form>

</div>

<div class="jensabox">
    <div class="fill-box">
    <div class="content-box">
        <h2>Do you really want to cancel this project?</h2>
        <p>This change will be irreversible.</p>
        <form class="u-pull-left" method="post" action="/api/changeProjectStatus">
            <input type="hidden" name="currentType" value="${project.getStatus()}">
            <input type="hidden" name="projectId" value="${project.getId()}">
            <button class="button button-red" type="submit" name="answer" value="cancelled">Cancel Project</button>
        </form>
        <a href="#" class="button button-cancel">No</a>
    </div>
    </div>
</div>

<script>
    $(".fancybox").fancybox();

    $('span.add_day').click(function() {
        setDays();
        $('span.add_day').css('display', 'none');
        $('select#day').css('display', 'block');
        $('select#day').addClass('visible');
    });

    $('select#month option').each(function(){
        var d = new Date();
        var m = d.getMonth();

        if($(this).val() < m+1) {
            $(this).remove();
        }
    });

    var searchNext = true;

    var removeNotMatches = function(list, q, sync) {
        var matches = [];
        var regex = new RegExp(q, "i");
        for (var i = 0; i < list.length; i++) {
            if (regex.test(list[i])) {
                if (list == pres) {
                    matches.push(list[i])
                } else if (pres.indexOf(list[i]) < 0) {
                    matches.push(list[i])
                }}}
        return matches;
    }


    var pres = ["Online advertising",
        "Billboard ad",
        "TV Promotion",
        "Face-to-face event",
        "Webinar",
        "Direct mail"];

    var preset = function(q, sync) {
        if(q == '')
            sync(pres)
        else
            sync(removeNotMatches(pres, q));
    }

    var types = [];

    var typesFilter = function(q, sync, async) {
        if(q.length == 3 && searchNext) {
            return $.ajax({
                dataType: "json",
                url: "/getTypes",
                data: {query: q},
                success: function(data) {
                    types = data;
                    async(removeNotMatches(types));
                }
            });
        } else
            return removeNotMatches(types, q, sync);
    }

    $('#type').typeahead({
                hint: false,
                highlight: true,
                minLength: 0
            }, {
                name: 'preset',
                source: preset
            }, {
                name: "type",
                source: typesFilter
            }

    ).change(function() {
                if($(this).val().length < 3)
                    searchNext = true;
                if($(this).val().length > 4)
                    searchNext = false;
            });

    $('select#month').change(function() {
        if($('select#day').hasClass('visible')){
            setDays();
        }
    });

    function setDays() {
        var days = 31;
        var e = document.getElementById("month");
        var strMonth = e.options[e.selectedIndex].value;

        if(strMonth == "1" || strMonth == "3" || strMonth == "5" ||
                strMonth == "7" || strMonth == "8" || strMonth == "10" || strMonth == "12") {
            days = 31;
        } else if(strMonth == "4" || strMonth == "6" || strMonth == "9" || strMonth == "11") {
            days = 30;
        } else if(strMonth == "2") {
            days = 28;
        }
```

```
        $('select#day').html("");
        for(var i=1; i<=days; i++) {
            $('select#day').append("<option value='" + i + "'>" + i +".</option>");
        }
    }
</script>
<script type="text/javascript" src="js/project-functions.js"></script>

</body>
</html>
```

Reset.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Andreas Poulsen
  Date: 28-Apr-15
  Time: 17:35
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <meta charset="utf-8">
    <title>Dell - Reset password</title>
    <meta name="description" content="Dell campaign management system">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href='http://fonts.googleapis.com/css?family=Roboto:400,300,500&subset=latin,latin-ext' rel='stylesheet'
type='text/css'>
    <link href="/css/normalize.css" rel="stylesheet" media="all">
    <link href="/css/skeleton.css" rel="stylesheet" media="all">
    <link href="/css/style.css" rel="stylesheet" media="all">
</head>
<body>
<div class="login-background u-pull-left">

    <div class="container" style="margin-top: 200px;">
        <div class="row">
            <div class="eight columns" style="padding-right: 30px;">
                <h1>Set password</h1>
                <p>Please set the password you will use to access <br/>Dell campaign management system.</p>
            </div>
            <div class="four columns login-form">
                <div class="message u-full-width u-pull-left">
                    <p><c:out value="${error}"></c:out></p>
                </div>
                <form action="/reset-password" method="post" class="u-full-width">
                    <input type="hidden" name="nonce" value="<c:out value="${param.n}"></c:out>">
                    <input type="hidden" name="user_id" value='<c:out value="${userId}"></c:out>'>
                    <input type="password" class="u-full-width just-line" name="pw" placeholder="Enter your new password">
                    <input type="submit" class="u-full-width button submit" value="Change Password">
                </form>
            </div>
        </div>
    </div>

</div>

</body>
</html>
```

Statistics.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<jsp:include page="header.jsp" />

<script type="text/javascript">

  // Load the Visualization API and the piechart package.
  google.load('visualization', '1.0', {'packages':['corechart','line', 'geochart']});

  // Set a callback to run when the Google Visualization API is loaded.
  google.setOnLoadCallback(drawChart);

  // Callback that creates and populates a data table,
  // instantiates the pie chart, passes in the data and
  // draws it.
  function drawChart() {
    var charts = ${statistics};
    for(var i = 0; i < charts.length; i++) {
      if (charts[i][0][1] == 'donut')
        drawDonut(charts[i]);
      else if(charts[i][0][1] == 'line')
        drawLine(charts[i]);
      else if(charts[i][0][1] == 'number')
        drawNumber(charts[i]);
      else if(charts[i][0][1] == 'geomap')
        drawMap(charts[i])
    }

  }
  function drawDonut(options) {
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Type');
    data.addColumn('number', 'Count');

    var d = [];
    for(var i = 1; i < options.length; i++) {
      d.push(options[i]);
      console.log(d);
    }
    data.addRows(d);

    // Set chart options
    var o = {'title':options[0][0],
      'backgroundColor':'transparent',
      'width':450,
      'height':300,
      'pieHole':0.4};

    var id = options[0][0].split(" ").join("");

    if(id == 'Typeswithhighestsuccessrate') {
      var formatter = new google.visualization.NumberFormat({pattern:'#%'});
      formatter.format(data, 1);
    }

    // Instantiate and draw our chart, passing in some options.
    var div = document.createElement('div');
    div.id = id;
    document.getElementById('graphs').appendChild(div);
    var chart = new google.visualization.PieChart(document.getElementById(id));
    chart.draw(data, o);
  }

 function drawMap(options) {
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Region');
    data.addColumn('number', 'Count');

    var d = [];
    for(var i = 1; i < options.length; i++) {
      d.push(options[i]);
      console.log(d);
    }
    data.addRows(d);

    // Set chart options
    var o = {
      region: 150,
      'width':800,
      'height':500};

    var id = options[0][0].split(" ").join("");


    // Instantiate and draw our chart, passing in some options.
    var div = document.createElement('div');
    div.id = id;
    var title = document.createElement('h5');
    title.innerText = options[0][0];
    document.getElementById('graphs').appendChild(title);
    document.getElementById('graphs').appendChild(div);
    var chart = new google.visualization.GeoChart(document.getElementById(id));
    chart.draw(data, o);
  }

  function drawLine(options) {
    var id = options[0][0].split(" ").join("");

    var data = new google.visualization.DataTable();
    data.addColumn('date', 'Date');
    data.addColumn('number', 'Budget Expenses');
    data.addColumn('number', 'Budget Projection');

    var d = [];
    var c = 0;

    var m = new Date(options[1][0]).getMonth(),
            q = ((m-1) / 3 ) + 1,
            y = new Date(options[1][0]).getFullYear(),
            sM= (q-1)*(12/4),
            eM= q*(12/4),
            sD= new Date(y, sM, 1),
            eD= new Date(y, eM, 0);
    for(var i = 1; i < options.length; i++) {
      if(id=='BudgetProgression') {
        options[i][0] = new Date(options[i][0]);
        c += options[i][1];
        options[i][1] = c;
        options[i][2] = null;
      }
      d.push(options[i]);
    }
    d.push([sD, null,0]);
    d.push([eD, null,${activeBudget.getInitial_budget()}]);
    console.log(d);
    data.addRows(d);

    // Set chart options
    var o = {
      title: options[0][0],
      backgroundColor: 'transparent',
      width:700,
      height:300,
      hAxis: {
        minValue: sD,
        maxValue: eD,
        format: 'MMM'
      },
      vAxis: {
        maxValue: ${activeBudget.getInitial_budget()}
      },
      series: {
        0: {
          lineWidth: 5
        },
        1: {
          lineWidth: 1,
          color: '#dc3912'
        }
      },
      trendlines: {
        0: {
          type: 'linear',
          color: 'green',
          lineWidth: 0,
          opacity: 0.3,
          visibleInLegend: true,
          labelInLegend: 'Trendline',
          pointSize: 2
        }
      }
    };


    // Instantiate and draw our chart, passing in some options.
    var div = document.createElement('div');
    div.id = id;
    document.getElementById('graphs').appendChild(div);
    var chart = new google.visualization.LineChart(document.getElementById(id));
    chart.draw(data, o);
  }

  function drawNumber(options) {
    console.log(options);

    var tr = document.createElement('tr');
    tr.innerHTML = '<td>' + options[0][0] + '</td><td>' + options[1][0] + '</td>';
    document.getElementById('numbersBody').appendChild(tr);
  }
</script>

<div class="container" style="padding-bottom: 30px;">
<div class="big-paper">

  <div class="u-pull-left">
    <h3>Statistics</h3>
    <p class="company-desc">Select quarter and see the graphical representation of projects and partners.</p>
  </div>

  <div class="quarter-selector">
    <form action="/statistics" method="get">
      <select name="quarter">
        <c:forEach items="${budgets}" var="budget">
          <option value="<c:out value="${budget.getYear()}Q${budget.getQuarter()}" />"><c:out
value="${budget.getYear()}Q${budget.getQuarter()}" /></option>
        </c:forEach>
      </select>
      <input type="submit" value="Go!">
    </form>
  </div>

  <div id="graphs">
  </div>
  <div id="numbers">
    <h5>Since the beginning of time</h5>
    <table class="u-full-width">
      <tbody id="numbersBody">
      </tbody>
    </table>
  </div>

</div>
</div>

</body>
</html>
```

User.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

  <jsp:include page="header.jsp" />

  <div class="container" style="margin-top: 30px; padding-bottom: 30px;">
      <div class="u-pull-right" style="margin-top: 10px;">
          <c:if test="${partner.getImg_filename() != null}" >
              <a href="partner?id=<c:out value='${partner.getId()}' />">
                <img class="company-image" style="margin-right: 0px;" src="/resources/companies/<c:out
value='${partner.getId()}' />/<c:out value='${partner.getImg_filename()}' />" />
              </a>
          </c:if>
      </div>

      <div class="cancel-box" style="margin-top: 10px; margin-right: 10px;"><a href="#" class="cancel-button jensabox-
trigger">Delete user</a></div>

      <h3 class="company-name"><c:out value='${user.getName()}' /></h3>
      <div class="u-pull-left" style="clear: left;">
          <p class="company-desc"><c:out value="${user.getEmail()}" /></p>
          <p class="company-desc">You can see all <c:out value='${user.getName()}' />'s projects below.</p>
      </div>

      <div class="table-head">
          <span class="id">ID</span>
          <span class="partner">Partner</span>
          <span class="type">Type</span>
          <span class="state">State</span>
          <span class="execution-date">Execution date</span>
      </div>
      <c:forEach var="project" items="${projects}">
          <a href="/project?id=<c:out value="${project.getId()}" />">
              <div class="project-item">
                  <span class="id"><strong>#</strong><c:out value="${project.getId()}" /></span>
                  <span class="partner"><c:out value='${partner.getName()}' /></span>
                  <span class="type"><c:out value="${project.getType()}" /></span>
                  <span class="state small"><c:out value="${project.getStatus()}" /></span>
                  <span class="execution-date small isShortDate"><c:out
value="${project.getExecution_date()}"></c:out></span>
```

```
            </div>
          </a>
      </c:forEach>

  </div>

<div class="jensabox">
    <div class="fill-box">
        <div class="content-box">
            <h2>Do you really want to delete this user?</h2>
            <p>This change will be irreversible.</p>
            <form class="u-pull-left" action="/markUserDeleted" method="post">
                <input type="hidden" value="<c:out value="${user.getId()}"></c:out>" name="viewedUser">
                <button class="button button-red" type="submit">Delete user</button>
                <a href="#" class="button button-cancel">No</a>
            </form>
        </div>
    </div>
</div>

  </body>
</html>
```

Users.jsp

```
<%@ page import="Domain.User" %>
<%@ page import="Domain.DisplayProject" %>
<%@ page import="java.util.ArrayList" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

  <jsp:include page="header.jsp" />

  <div class="container" style="margin-top: 30px; padding-bottom: 30px;">
      <a href="/create-user" class="button-clear u-pull-right">Add user</a>
      <h3 style="margin-bottom: 1rem;">Users</h3>
      <p style="color: #9F9F9F;">Click on user to get to the user's details.</p>

      <div class="table-head">
          <span class="id">ID</span>
          <span class="user-name">Name</span>
          <span class="email">Email</span>
      </div>
      <c:forEach var="user" items="${users}">
          <a href="/user?id=<c:out value='${user.getId()}' />">
              <div class="project-item">
                  <span class="id"><strong>#</strong><c:out value="${user.getId()}" /></span>
                  <span class="user-name"><c:out value="${user.getName()}" /></span>
                  <span class="email"><c:out value="${user.getEmail()}" /></span>
              </div>
          </a>
      </c:forEach>
  </div>

  </body>
</html>
```