TEST CASE FOR USE CASE 12

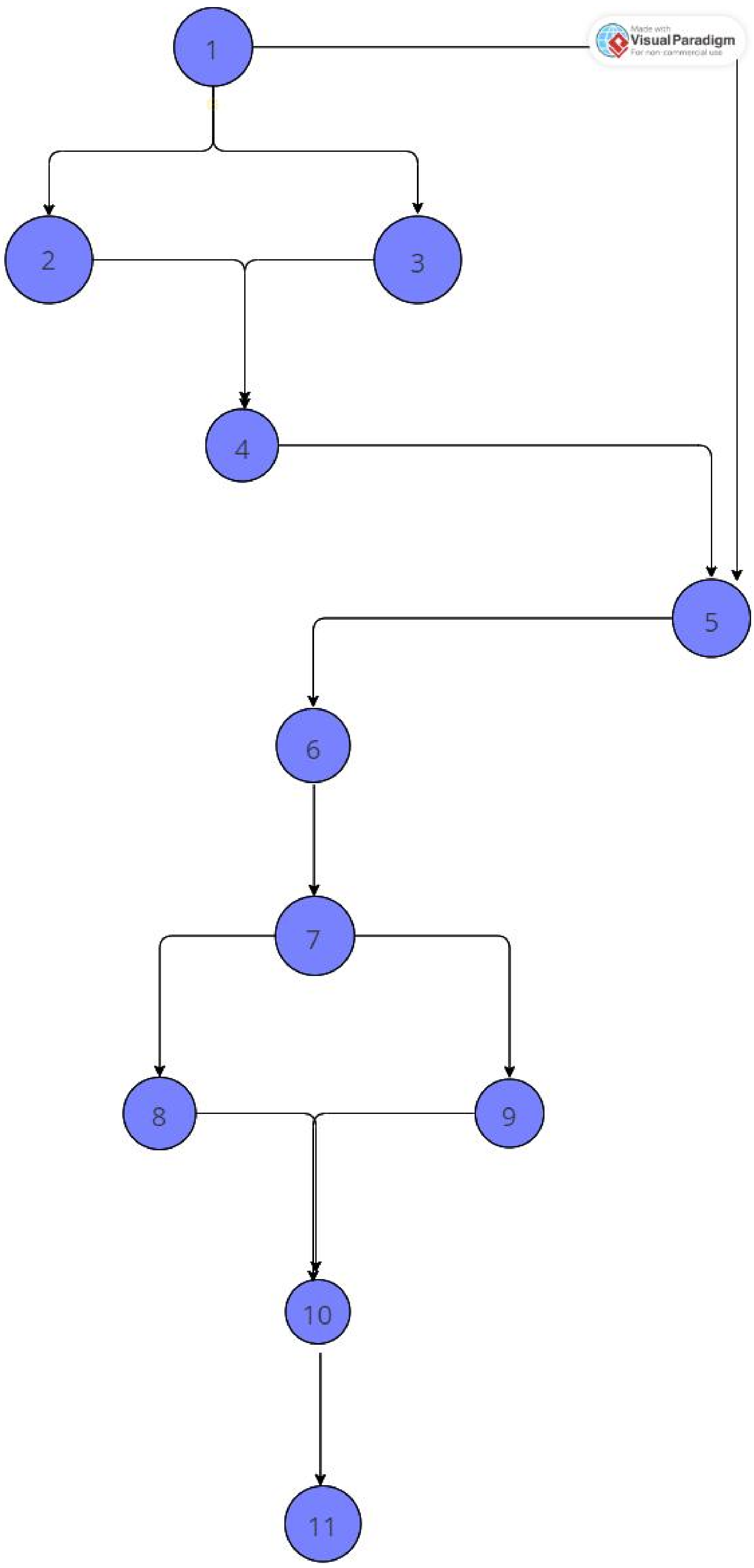
```
private void populateRequests() {
   try {
      // Establish connection to your database
      Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/softengin23_24",
"root", "W45@jqr#8CX");

      // Prepare the SQL query
      String query = "SELECT * FROM requests"; // Replace 'requests' with the actual name of your table
      PreparedStatement preparedStatement = connection.prepareStatement(query);


      // Execute the query and obtain the result set                        \ 1\
      ResultSet resultSet = preparedStatement.executeQuery();

      // Create a DefaultListModel to hold the data
      DefaultListModel<String> listModel = new DefaultListModel<>();

      // Iterate over the result set and add each row to the list model
                    2
        while (resultSet.next()) {

         int requestId = resultSet.getInt("idrequests");
         String username = resultSet.getString("username");
         String name = resultSet.getString("name");
         String date = resultSet.getString("date");
         String about = resultSet.getString("about");
         String userType = resultSet.getString("user_type");
3


         String rowData = String.format("Username: %s, Date: %s" ,
               username, date);

         // Add the row's data to the list model
         listModel.addElement(rowData);
      }

      // Set the populated list model to the requestList JList
      requestList.setModel(listModel);


       // Close resources                              4
      resultSet.close();
      preparedStatement.close();
      connection.close();
   } catch (SQLException ex) {

      ex.printStackTrace(); // Handle any potential errors here        5
   }
}
```

```
private void handleSendToAdmin() {
   String username = textField1.getText(); // Get the username from textField1          \1\
   String reason = textField2.getText(); // Get the reason from textField2

   // Check if either of the text fields is empty
              2                    3
if (username.isEmpty() || reason.isEmpty()) {
      JOptionPane.showMessageDialog(this, "Missing information. Please fill out all fields.",
"Missing Information", JOptionPane.ERROR_MESSAGE);                                    4
      return; // Exit the method if validation fails      5
   }

   try {
      // Establish connection to your database                          6
      Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/softengin23_24", "root",
"W45@jqr#8CX");

      // Prepare the SQL INSERT query
      String insertQuery = "INSERT INTO emergencies (username, reason) VALUES (?, ?)";
      PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
      preparedStatement.setString(1, username);
      preparedStatement.setString(2, reason);

      // Execute the INSERT query
      int rowsAffected = preparedStatement.executeUpdate();

      // Check if the insertion was successful

          7
    if (rowsAffected > 0) {
         JOptionPane.showMessageDialog(this, "Emergency sent to admin!", "Success",
JOptionPane.INFORMATION_MESSAGE);                         8
      } else {
         JOptionPane.showMessageDialog(this, "Failed to send to admin", "Error",
JOptionPane.ERROR_MESSAGE);                           9
```

```
        }

        // Close resources
        preparedStatement.close();
        connection.close();                                 10
    } catch (SQLException ex) {
        ex.printStackTrace(); // Handle any potential errors here        11
        JOptionPane.showMessageDialog(this, "Error saving emergency to database",
"Database Error", JOptionPane.ERROR_MESSAGE);
    }
}




private void loadRequestsData() {
    try {
        // Establish connection to your database
        Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/softengin23_24", "root",
"W45@jqr#8CX");

        // Prepare the SQL query
        String query = "SELECT * FROM requests"; // Replace 'requests' with the actual name of
your table
        Statement statement = connection.createStatement();

        // Execute the query and obtain the result set
        ResultSet resultSet = statement.executeQuery(query);
\1\
        // Get the metadata of the result set
        ResultSetMetaData metaData = resultSet.getMetaData();

        // Number of columns in the result set
        int columnCount = metaData.getColumnCount();

        // Column names
        Vector<String> columnNames = new Vector<>();

                                2
for (int column = 1; column <= columnCount; column++) {
            columnNames.add(metaData.getColumnName(column));    3
        }

        // Data of the table
        Vector<Vector<Object>> data = new Vector<>();        4
```

```
                    5
         while (resultSet.next()) {
         Vector<Object> vector = new Vector<>();        6
                                                        7
        for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
            vector.add(resultSet.getObject(columnIndex));  8
         }
         data.add(vector);   9
      }

      // Close resources
      resultSet.close();
      statement.close();
      connection.close();                    10

      // Set the table model
      DefaultTableModel model = new DefaultTableModel(data, columnNames);
      table1.setModel(model);
   } catch (SQLException ex) {
      ex.printStackTrace(); // Handle any potential errors here
      JOptionPane.showMessageDialog(this, "Error fetching data from database", "Database
Error", JOptionPane.ERROR_MESSAGE);   11
   }
}
```

Για τη συνάρτηση populateRequsts() και loadRequestsData() δεν χρειάστηκε κάποια
απαρίθμηση στη ροή του κώδικα καθώς δεν υπάρχει κάποιο statement που
να αλλάζει τη ροή του κώδικα.
Παρακάτω παρατίθεται ο Γράφος για την συνάρτηση handleSendToAdmin().


1
2
3
4
5
6
7
8
9
10
11
Made with Visual Paradigm For non-commercial use

Σύμφωνα με τον τύπο: V(g) = e-n+2 = 14-11+2 = 5 περιοχές, όπως φαίνεται και στον γράφο.

Εφαρμόζουμε τον αλγόριθμο για εύρεση μονοπατιών.
Ξεκινάμε με το συντομότερο
έγκυρο μονοπάτι:

1-5-6-7-8-10-11 ή 1-5-6-7-9-10-11

Άρα Μ1: 1-5-6-7-8-10-11 ή (Main flow στο Use Case)
Μ1: 1-5-6-7-8-10-11
Ακολουθούν τα υπόλοιπα μονοπάτια που προκύπτουν με προσθήκη ακμών στο
Μ1.
Μ2: 1-2-4-5-6-7-8-10-11 ή
Μ2: 1-3-4-5-6-7-8-10-11

Μ3: 1-2-4-5-6-7-9-10-11 ή
Μ3: 1-3-4-6-7-8-10-11

Path	Περιγραφή	Περίπτωση Ελέγχου	Αναμενόμενο Αποτέλεσμα (Έξοδος Προγράμματος)
Μ1	ΑΠΟΣΤΟΛΗ REQUEST ΣΤΟΝ ADMIN ΓΙΑ AUTHORIZE	handleSentToAdmin()	"Emergency sent to admin!"
Μ2	ΕΛΕΓΧΟΣ ΓΙΑ ΠΛΗΡΟΦΟΡΙΕΣ	username.isEmpty() reason.isEmpty()	"Missing information. Please fill out all fields."
Μ3	ΕΛΕΓΧΟΣ ΓΙΑ ΑΠΟΣΤΟΛΗ	rowsAffected > 0	"Failed to send to admin"