Software Plan

**1. Scope:**

1.1. Functions: The phone Voting System would be used by two types of users. An administrator who would start the application and allow voting to start and stay running until end of voting where he would terminate the ballot and calculate the votes to announce the winner poster. The other user type is a voter who would register with the software to allow him to submit a vote.

1.2. Performance: This system should handle a one-way communication between the voters and the administrator. The system should be able to handle 1024 concurrent users registered on the database at any given time.

1.3. Limitations: This system should only accept votes from users who are registered, however there is not restriction on who is allowed to register as long as they are not registering twice. The system also should allow any given user to vote only once, however if a user submits a vote, they will not be able to overturn that vote.

**2.Tasks:**

2.1. User Interface: Users will be able to send a message through the application to register a phone number. Users will also be able to send number-coded messages to share a vote, or to check on status of voting.

2.2. Database Management: User will interact with the system by sending phone messages in the format of a number code. Each number code represents a state condition as outlined below. When the user sends a code message, he would receive a message back notifying him of the result to his query. The submitted user response will be entered into a database locally to be displayed later for evaluation.

**3. Resources:**

3.1. Hardware: The system should run on an android mobile device, or on a computer with installed Android Studio for simulation. Users will be expected to have a mobile device with the application download and installed.

3.2. Software: The Phone Voting System is developed using Android Studio and is written using Java.

3.3. People: Users are expected to be either students at University of Pittsburgh, or visitors on the School.

**4. Data Structure:**

The vote submitted by the user would be stored into a HashTable to ensure accepting only user IDs that have not voted before, thereby preventing duplicate submissions.

The description below shows the possible messages sent by users, and the translation of these messages into actions that will cause a state change.

**Variables**:

- **Passcode**: ****
- **SecurityLevel**: 3
- **Name**: VotingSoftware (Name of created Component)
- **SourceCode**: VS.jar (Source code file name of created Component)
- **InputMsgID 1**: 701 (CastVote with attributes VoterPhoneNo, CandidateID)
- **InputMsgID 2**: 702 (RequestReport with attributes Passcode,N)
- **InputMsgID 3**: 703 (Initialize TallyTable with attributes Passcode,CandidateList)
- **OutputMsgID 1**: 711 (AcknowledgeVote with attribute Status)
- **OutputMsgID 2**: 712 (AcknowledgeRequestReport with attribute RankedReport)
- **OutputMsgID 3**: 26 (Acknowledgement with attributes AckMsgID,YesNo,Name)
- **Component Description**: VotingSoftware checks voters using VoterTable and counts votes into TallyTable. Each voter is supposed to have a unique VoterPhoneNo.

  When message 701 is received, the VoterPhoneNo is checked against the VoterTable. If it is already there, the voter is prevented from voting, and message 711 is sent with Status 1 (duplicate vote). If it is not there, the VoterPhoneNo is added to VoterTable, and the CandidateID is checked against TallyTable. If CandidateID is not there, Message 711 is sent with Status 2 (invalid vote). If CandidateID is there, that entry is incremented and Message 711 is sent with Status 2 (valid vote).

  When message 702 is received and Passcode is verified, a RankedReport containing the first N CandidateID's with highest rankings is generated, and a message 712 is sent containing this RankedReport as a long string. If passcode is wrong, this RankedReport is null. At the end both VoterTable and TallyTable are destroyed.

  When message 703 is received and Passcode is verified, TallyTable is initialized by loading it with CandidateList. A general acknowledge message 26 is sent.
- **KnowledgeBase**: VoterTable,TallyTable

**MsgID**:22 **Description**: Kill Component **Example**:Msg22.XML

**Variables**:

- **Passcode** (Passcode of Administrator)
- **SecurityLevel** (Security Level of Administrator)
- **Name** (Name of killed Component)
- **SourceCode** (Source code file name of killed Component)

**MsgID**:23 **Description**: Connect to server **Example**:Msg23.XML

**Variables**:

- **Passcode** (Passcode of Administrator)
- **SecurityLevel** (Security Level of Administrator)
- **Name** (Name of Component to be connected to server)

**MsgID**:24 **Description**: Activate Component **Example**:Msg24.XML

**Variables**:

- **Passcode** (Passcode of Administrator)
- **SecurityLevel** (Security Level of Administrator)
- **Name** (Name of activated Component)
- **SourceCode** (Source code file name of activated Component)

**MsgID**:25 **Description**: DeActivate Component **Example**:Msg25.XML

**Variables**:

- **Passcode** (Passcode of Administrator)
- **SecurityLevel** (Security Level of Administrator)
- **Name** (Name of deactivated Component)
- **SourceCode** (Source code file name of activated Component)

**MsgID**:26 **Description**: Acknowledgement **Example**:Msg26.XML

**Variables**:

- **AckMsgID** (This is Ack for MsgID)
- **YesNo** (Yes/No for positive/negative Ack)
- **Name** (Name of involved Component)

**MsgID**:701 **Description**: Cast Vote **Example**:Msg701.XML

Variables:

- **VoterPhoneNo**
- **CandidateID**

**MsgID:**702 **Description**: Request Report **Example**:Msg702.XML

Variables:

- **Passcode**
- **N** (number of winners)

**MsgID:**703 **Description**: Initialize Tally Table **Example**:Msg703.XML

Variables:

- **Passcode**
- **CandidateList** (a string of CandidateID separated by semicolons)

**MsgID:**711 **Description**: Acknowledge Vote **Example**:Msg711.XML

Variables:

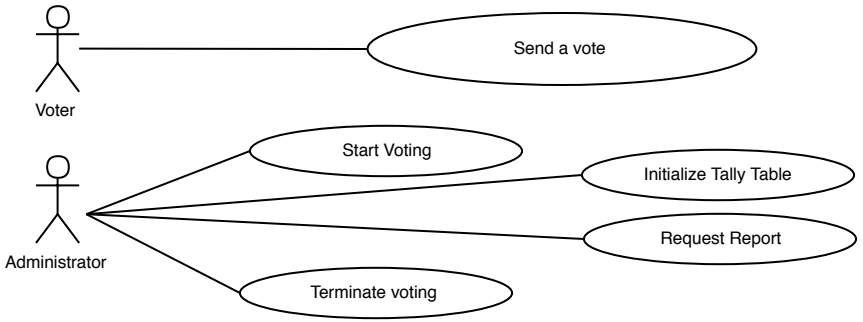
- **Status** (1: duplicate, 2: invalid, 3: valid)

**MsgID:**712 **Description**: Acknowledge RequestReport **Example**:Msg712.XML

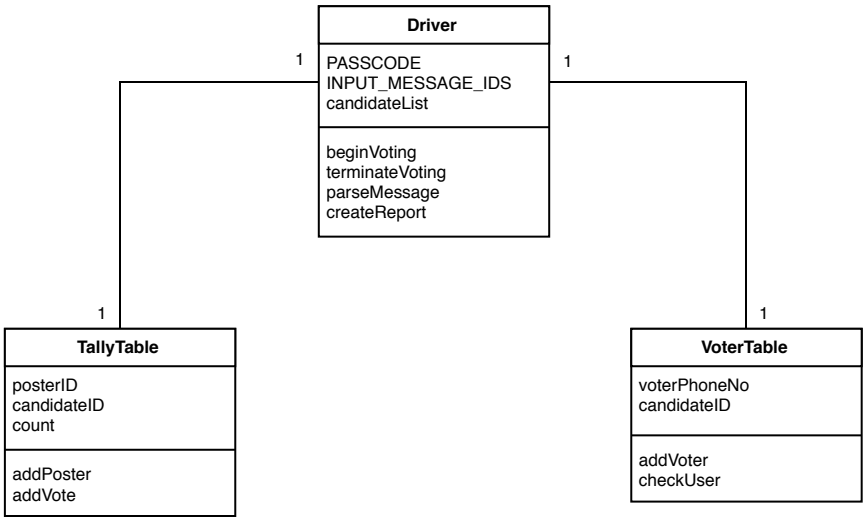
Variables:

- **RankedReport** (a string of N CandidateID,NumberVote separated by semicolons)

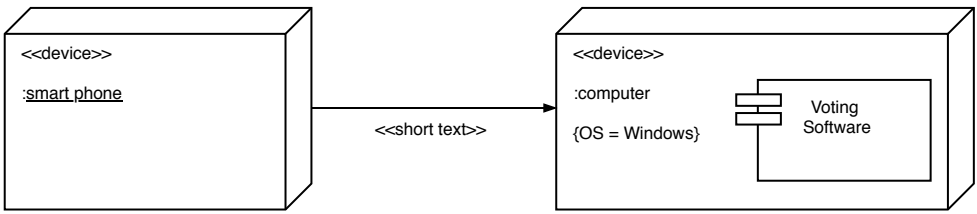
# Use Case Diagram



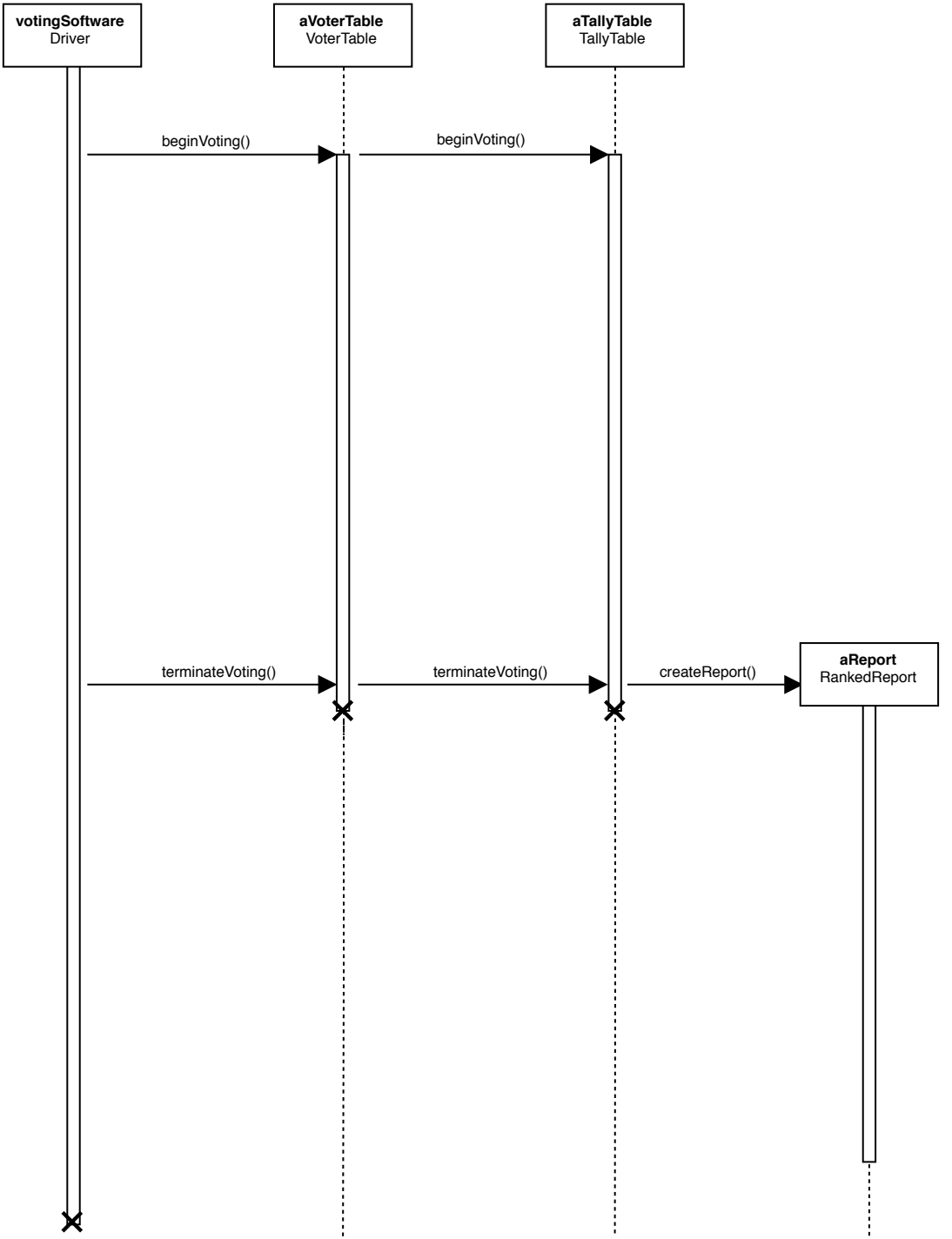

# Class Diagram



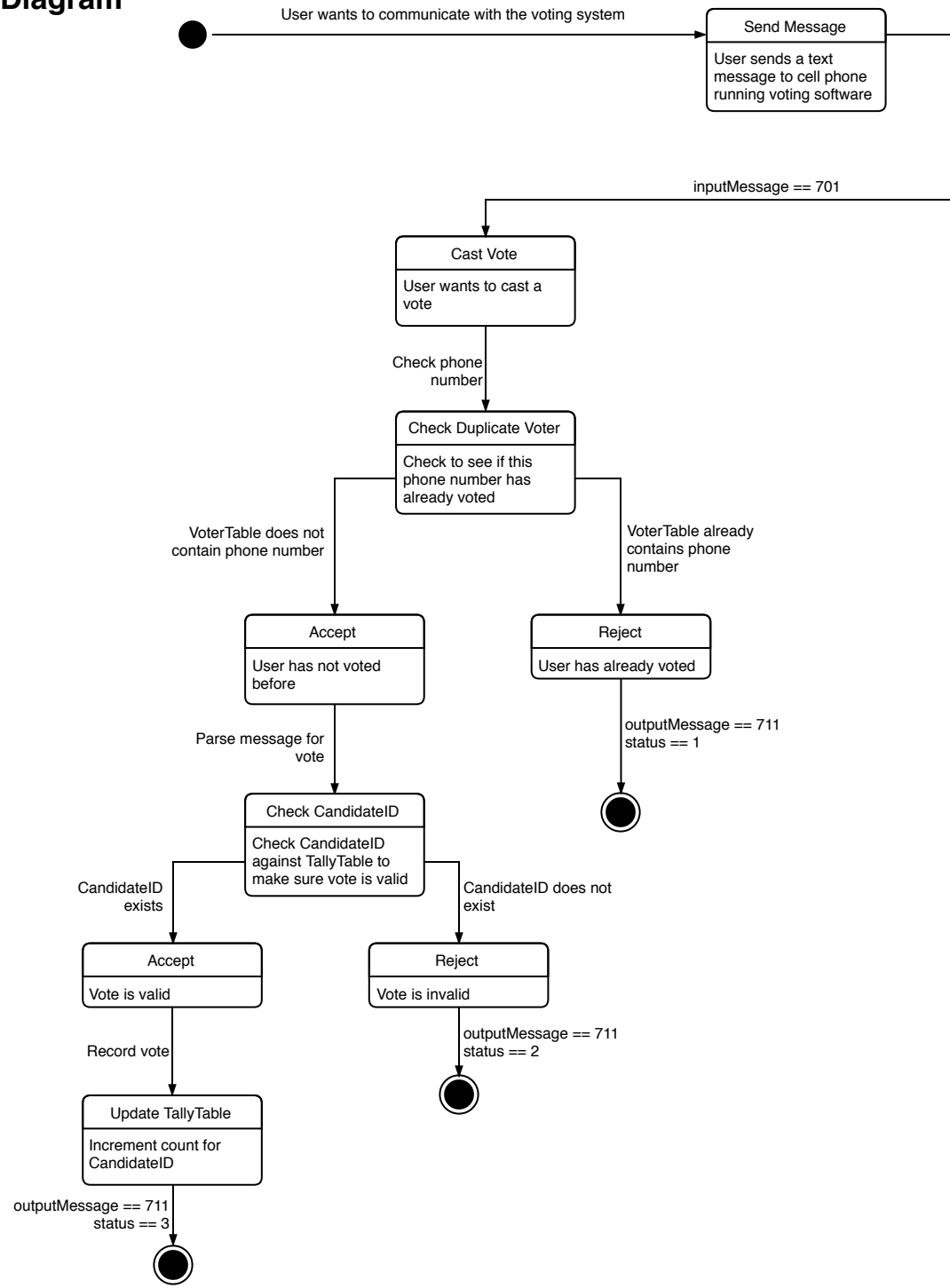

# Deployment Diagram

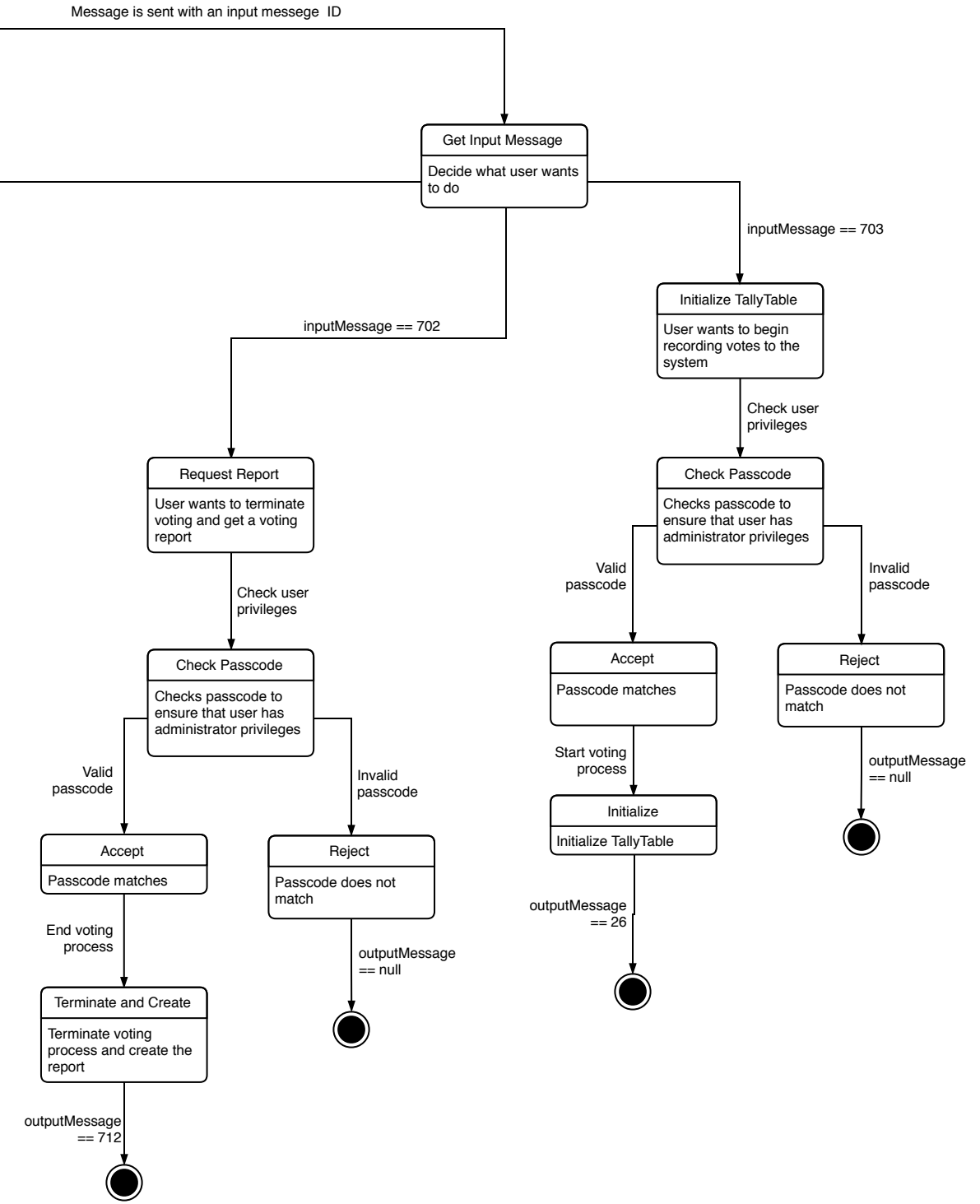


# Sequence Diagram



# State Diagram




Message is sent with an input messege ID
Get Input Message
Decide what user wants to do
inputMessage == 703
Initialize TallyTable
User wants to begin recording votes to the system
Check user privileges
Check Passcode
Checks passcode to ensure that user has administrator privileges
Valid passcode
Invalid passcode
Accept
Passcode matches
Reject
Passcode does not match
outputMessage == null
Start voting process
Initialize
Initialize TallyTable
outputMessage == 26
inputMessage == 702
Request Report
User wants to terminate voting and get a voting report
Check user privileges
Check Passcode
Checks passcode to ensure that user has administrator privileges
Valid passcode
Invalid passcode
Accept
Passcode matches
Reject
Passcode does not match
outputMessage == null
End voting process
Terminate and Create
Terminate voting process and create the report
outputMessage == 712