3.1

```
private void openFile() {
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showOpenDialog(this);

    if (result == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            StringBuilder content = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                Content+ line +"\n";
            }
            plainTextArea.setText(content.toString());

        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Error reading file: " + ex.getMessage(),
                                        "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

3.2

```
    private void encryptMessage() {
        String text =plainTextArea.getText();
        if (text.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No text to encrypt!",
                                        "Warning", JOptionPane.WARNING_MESSAGE);
            return;
        }

        String encryptedText = encrypt(text);
        encryptedTextArea.setText(encryptedText);
        JOptionPane.showMessageDialog(this, "Message encrypted successfully!",
                                    "Success", JOptionPane.INFORMATION_MESSAGE);
    }

    private String encrypt(String text) {
        StringBuilder encrypted = new StringBuilder();


        String lowerCaseText = text.toLowerCase();

        for (char c : lowerCaseText.toCharArray()) {
            if (Character.isLetter(c)) {

                char encryptedChar = (char) (((c - 'a' + 3) % 26) + 'a');
                encrypted.append(encryptedChar);
            } else {

                encrypted.append(c);
            }
        }

        return encrypted.toString();
    }


3.4 private void saveToDatabase() {
    String encryptedText = encryptedTextArea.getText();

    if (encryptedText.isEmpty()) {
        JOptionPane.showMessageDialog(this, "No encrypted text to save!",
                                    "Warning", JOptionPane.WARNING_MESSAGE);
        return;
    }

    Connection connection = null;

    try {

        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

        String dbURL = "jdbc:derby:messageDB;create=true";
        connection = DriverManager.getConnection(dbURL);




        String insertSQL = "INSERT INTO messages (encrypted_text, timestamp) VALUES (?, ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(insertSQL)) {
            pstmt.setString(1, encryptedText);


            String timestamp = new Date();
            pstmt.setString(2, timestamp);


            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(this,
                    "Encrypted message saved to database successfully!\n" +
                    "Timestamp: " + timestamp,
                    "Success", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "Failed to save to database",
                                            "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

    } catch (ClassNotFoundException ex) {
        JOptionPane.showMessageDialog(this,
            "Database driver not found: " + ex.getMessage(),
            "Database Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this,
            "Error saving to database: " + ex.getMessage(),
            "Database Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    } finally {

        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

3.5 private void retrieveFromDatabase() {

```
    Connection connection = null;
    try {

        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");


        String dbURL = "jdbc:derby:messageDB;create=true";
        connection = DriverManager.getConnection(dbURL);

        String selectSQL = "SELECT id, encrypted_text, timestamp FROM messages ORDER BY
timestamp DESC";

        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(selectSQL)) {

            StringBuilder allMessages = new StringBuilder();
            int messageCount = 0;

            while (rs.next()) {
                messageCount++;
                int id = rs.getInt("id");
                String encryptedText = rs.getString("encrypted_text");
                String timestamp = rs.getString("timestamp");


                allMessages.append("=== Message ID: ").append(id).append(" ===\n");
                allMessages.append("Time: ").append(timestamp).append("\n");
                allMessages.append("Encrypted Text:\n");
                allMessages.append(encryptedText).append("\n");
                allMessages.append("=".repeat(40)).append("\n\n");
            }

            if (messageCount > 0) {
                encryptedTextArea.setText(allMessages.toString());
                JOptionPane.showMessageDialog(this,
                    "Retrieved " + messageCount + " message(s) from database",
                    "Success", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this,
                    "No messages found in database",
                    "Info", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    } catch (ClassNotFoundException ex) {
        JOptionPane.showMessageDialog(this,
            "Database driver not found: " + ex.getMessage(),
            "Database Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this,
            "Error retrieving from database: " + ex.getMessage(),
            "Database Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    } finally {

        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

3.6

```
private void clearMessage() {
 plainTextArea.setText("");
    encryptedTextArea.setText("");
}
```

3.7

```
exitButton.addActionListener(e -> System.exit(0));
```