



Cappalog App Store

By: Tim Romer, Tianze Shan, Miles Ferstel, Robert Szymkowicz, Chester Lindner



Team Roles

Data Layer: Tim Romer

Developers: Chester Lindner & Miles Ferstel

Testers: Robert Szymkowicz & Tianze Shan

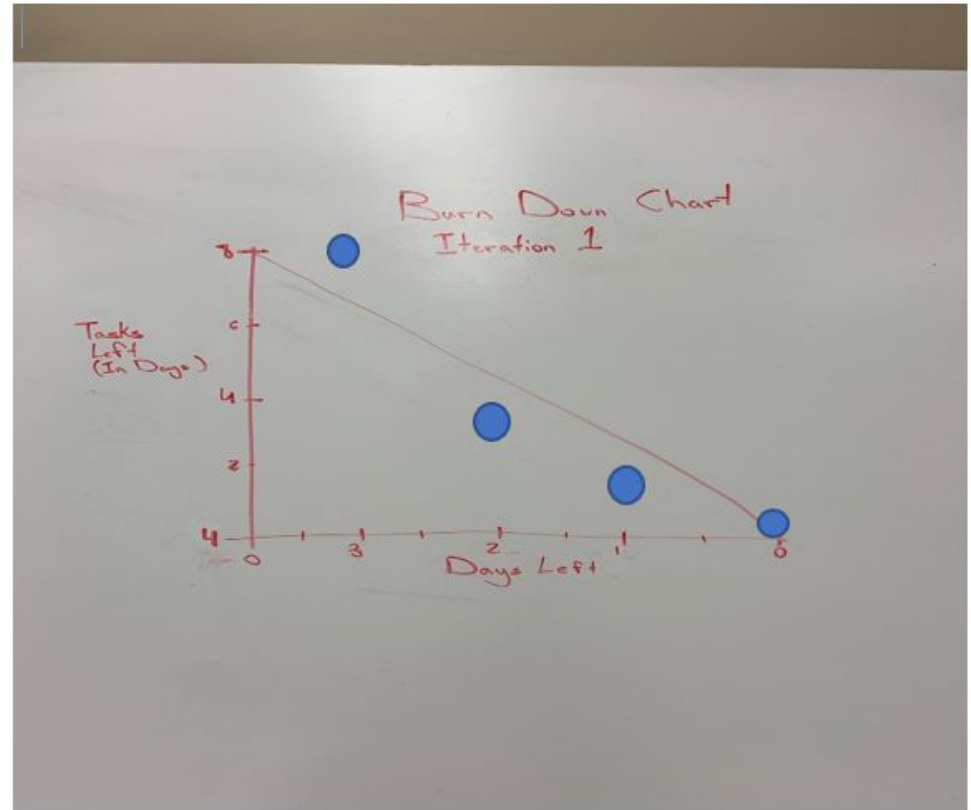
Project Manager: Tim Romer

Technical Manager: Robert Szymkowicz

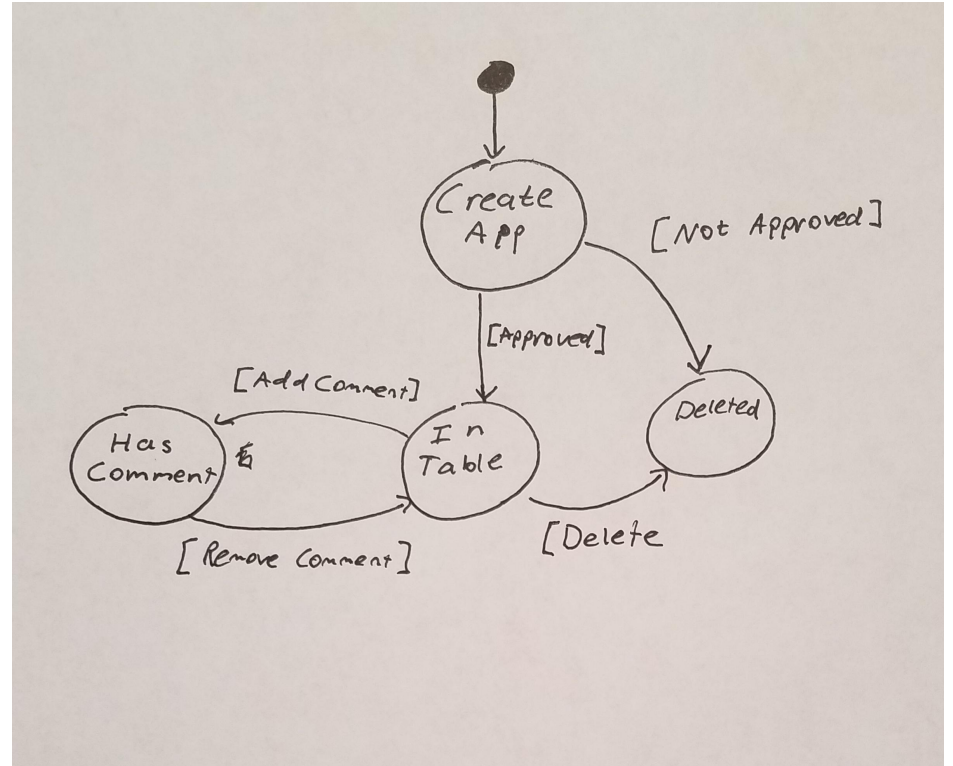
Burndown Chart

Planned tasks for Iteration 1 were completed on schedule.

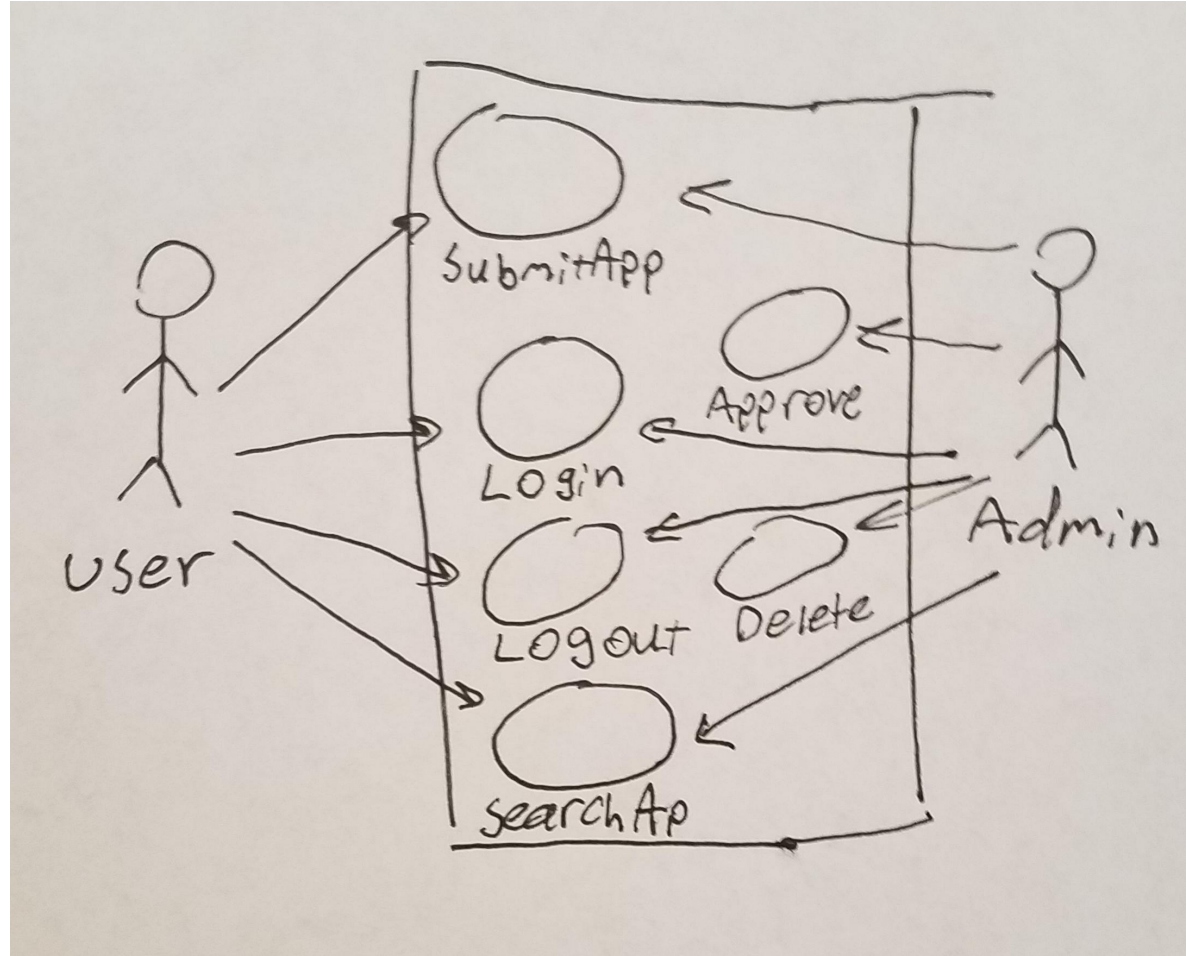
Velocity: Since



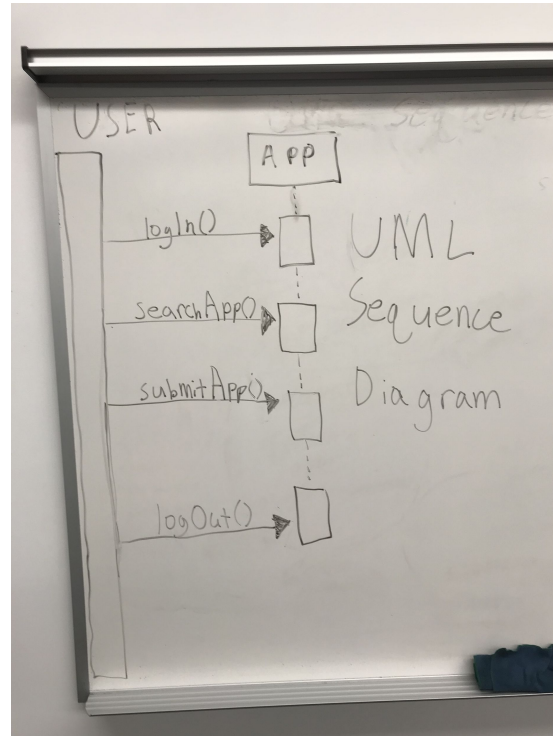
State Diagram: App



Use case



Sequence chart



```

1 import java.util.ArrayList;
2 import java.util.Map;
3
4 public class APP {
5     //=====properties
6     private String name, developer, link, description, platform;
7     private double price;
8     private Map<User, ArrayList<String>> comments;
9     //=====constructors
10    public APP(String name, String developer, String link, String description, String platform, double price) {
11        this.name = name;
12        this.developer = developer;
13        this.link = link;
14        this.description = description;
15        this.platform = platform;
16        this.price = price;
17        this.comments = null;
18    }
19
20    //=====getter/setter
21    public String getName() { return name; }
22
23    public void setName(String name) { this.name = name; }
24
25    public String getDeveloper() { return developer; }
26
27    public void setDeveloper(String developer) { this.developer = developer; }
28
29    public String getLink() { return link; }
30
31    public void setLink(String link) { this.link = link; }
32
33    public String getDescription() { return description; }
34
35    public void setDescription(String description) { this.description = description; }
36
37    public String getPlatform() { return platform; }
38
39    public void setPlatform(String platform) { this.platform = platform; }

```

```

1 public abstract class User {
2     //=====properties
3     private String userName;
4     private String password;
5
6     //=====constructors
7     public User(String userName, String password) {
8         this.userName = userName;
9         this.password = password;
10    }
11
12    public User() {
13        this.userName = null;
14        this.password = null;
15    }
16
17    //=====methods
18    public void request(APP a) {
19        //default
20    }
21
22    public boolean login(String userName, String password) {
23        //default
24        return false;
25    }
26
27    public boolean logout() {
28        //default
29        return false;
30    }
31
32    @Override
33    public String toString() {
34        return "User{" +
35            "userName='" + userName + '\'' +
36            ", password='" + password + '\'' +
37            '}';
38    }
39

```

```

1 public class Admin extends Moderator {
2     //=====constructors
3
4     public Admin(String userName, String password) {
5         super(userName, password);
6     }
7
8     public Admin() {
9     }
10
11    //=====methods
12    public void approve(APP a) {
13    }
14
15    public void deny(APP a) {
16    }
17
18    @Override
19    public void deletePost(String comment) { super.deletePost(comment); }
20
21    @Override
22    public void request(APP a) { super.request(a); }
23
24    @Override
25    public boolean login(String userName, String password) { return super.login(userName, password); }
26
27    @Override
28    public boolean logout() { return super.logout(); }
29
30    @Override
31    public String toString() { return super.toString(); }
32
33

```

```

1 public class Moderator extends User {
2     //=====constructors
3
4     public Moderator(String userName, String password) { super(userName, password); }
5
6     public Moderator() {
7     }
8
9     //=====methods
10    public void deletePost(String comment) {
11    }
12
13    @Override
14    public void request(APP a) { super.request(a); }
15
16    @Override
17    public boolean login(String userName, String password) { return super.login(userName, password); }
18
19    @Override
20    public boolean logout() { return super.logout(); }
21
22    @Override
23    public String toString() { return super.toString(); }
24

```

SQL Connection



```
public class SQLConnection {

    public static void main(String[] args) throws ClassNotFoundException {
        String url = "jdbc:mysql://localhost:3306/cappalogdb"; //databaseName=applist;u
        Connection connect = null;
        ArrayList<APP> appList = new ArrayList<APP>();
        try{
            Class.forName("com.mysql.jdbc.Driver");
            connect = DriverManager.getConnection(url,"romerta", "romert2GE");
            Statement stmt = connect.createStatement();
            String sql = "Select * From apptable;";
            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                int id = rs.getInt("appID");
                String name = rs.getString("Name");
                String dev = rs.getString("Developer");
                String link = rs.getString("Link");
                String desc = rs.getString("Description");
                String plat = rs.getString("Platform");
                double price = rs.getDouble("Price");
                APP app = new APP(id, name, dev, link, plat, plat, price);
                appList.add(app);
            }
            System.out.println(rs);

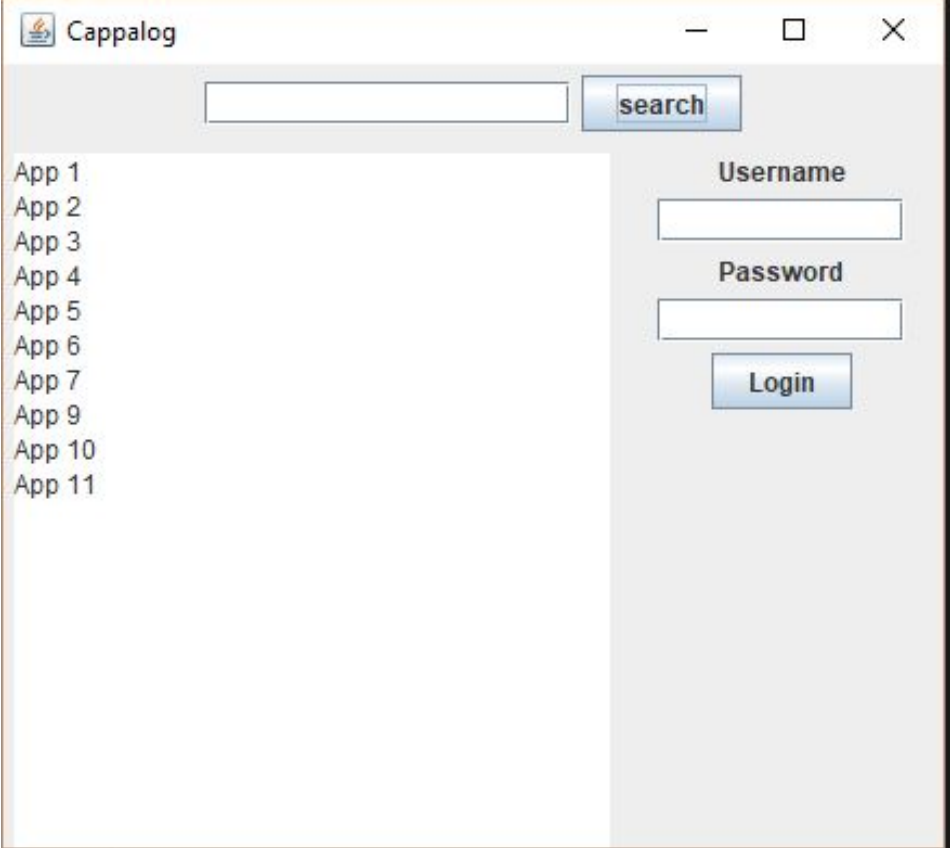
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        finally {
            if (connect != null) {
                try {
                    connect.close ();
                    System.out.println ("Database connection terminated");
                }
                catch (Exception e) { /* ignore close errors */ }
            }
        }
    }
}
```



```
2 • SELECT * FROM apptable;
```

[illegible]

Iteration 1 UI



The image shows a window titled "Cappalog" with standard window controls (minimize, maximize, close). The interface is divided into two main sections. The left section contains a list of application names: App 1, App 2, App 3, App 4, App 5, App 6, App 7, App 9, App 10, and App 11. The right section contains a search bar at the top with a "search" button. Below the search bar is a login form with labels for "Username" and "Password", each followed by an input field, and a "Login" button at the bottom.

Cappalog

search

App 1
App 2
App 3
App 4
App 5
App 6
App 7
App 9
App 10
App 11

Username

Password

Login

UI source Code

```
1 import java.awt.BorderLayout;
2 import java.awt.event.ActionEvent;
3 import java.awt.event.ActionListener;
4 import java.util.ArrayList;
5 import javax.swing.JButton;
6 import javax.swing.JFrame;
7 import javax.swing.JLabel;
8 import javax.swing.JPanel;
9 import javax.swing.JPasswordField;
10 import javax.swing.JTextArea;
11 import javax.swing.JTextField;
12
13 public class Cappalog extends JFrame {
14
15     private static final long serialVersionUID =
16     private final int FRAME_WIDTH = 450;
17     private final int FRAME_HEIGHT = 400;
18
19     private JTextField searchTextField;
20     private JButton searchButton;
21     private JTextArea wordListArea;
22     private JLabel userNameLabel;
23     private JLabel passwordLabel;
24     private JButton loginButton;
25
26     private JLabel userNameLabel;
27     private JLabel passwordLabel;
28     private JButton loginButton;
29     private JTextField unField;
30     private JPasswordField p1;
31
32     private ArrayList<String> wordList;
33
34     public Cappalog(){
35         wordList = new ArrayList<String>();
36         loadWordList();
37         this.setSize(FRAME_WIDTH,FRAME_HEIGHT);
38         this.getContentPane().add(createNorthPanel() , BorderLayout.NORTH);
39         this.getContentPane().add(createCenterPanel(), BorderLayout.CENTER);
40         this.getContentPane().add(createWestPanel(), BorderLayout.WEST);
41         search();
42     }
43
44     private void loadWordList(){
45         wordList.add("App 1");
46         wordList.add("App 2");
47         wordList.add("App 3");
48         wordList.add("App 4");
49         wordList.add("App 5");
50
51     }
52
53     private JPanel createNorthPanel(){
54         JPanel panel = new JPanel();
55         searchTextField = new JTextField(15);
56         panel.add(searchTextField);
57         searchButton = new JButton("search");
58         searchButton.addActionListener(new SearchButtonListener());
59         panel.add(searchButton);
60         return panel;
61     }
62
63     class SearchButtonListener implements ActionListener{
64         public void actionPerformed(ActionEvent event){
65             search();
66         }
67     }
68
69     private void search(){
70         String query = searchTextField.getText();
71         wordListArea.setText("");
72         if ( query.length() == 0){
73             for ( String word: wordList)
74                 wordListArea.append(word + "\n");
75         }
76     }
77 }
```



Retrospective

- What went well: Team gets along very well and everyone seems to have technical grasp of their roles
- What didn't go well: The communication could be improved
- What have we learned: Software engineering principles, GIT, and working in effectively groups
- What still puzzles me: Remote MySQL server connections



Iteration 2 Plans

- App submission
 - Users shall submit apps
 - Admins shall be able to view, delete, or approve request
 - Users should be able to search for any app by name