

```

1  package ballmerpeak.turtlenet.client;
2
3  import ballmerpeak.turtlenet.shared.CommentDetails;
4  import ballmerpeak.turtlenet.shared.PostDetails;
5  import ballmerpeak.turtlenet.shared.Message;
6  import ballmerpeak.turtlenet.shared.Conversation;
7
8  import com.google.gwt.core.client.*;
9  import com.google.gwt.event.dom.client.*;
10 import com.google.gwt.user.client.ui.*;
11 import com.google.gwt.event.logical.shared.*;
12 import com.google.gwt.user.client.Window;
13 import com.google.gwt.user.client.rpc.AsyncCallback;
14 import com.google.gwt.dom.client.Style.FontWeight;
15 import com.google.gwt.dom.client.DivElement;
16 import com.google.gwt.dom.client.Document;
17 import com.google.gwt.user.client.Timer;
18 import com.google.gwt.user.client.Window;
19 import java.util.Date;
20
21 public class frontend implements EntryPoint, ClickListener {
22
23     // Create remote service proxy to talk to the server-side Turtlenet service
24     private final TurtlenetAsync turtlenet = GWT.create(Turtlenet.class);
25     //private final TurtlenetAsync msgfactory = GWT.create(MessageFactory.class);
26     public void onModuleLoad() {
27         // Remove loading indicator from frontend.html
28         DivElement loadingIndicator = DivElement.as(Document.get().getElementById("loading"));
29         loadingIndicator.setInnerHTML("");
30
31         /* Add handler for window closing */
32         Window.addCloseHandler(new CloseHandler<Window>() {
33             public void onClose(CloseEvent<Window> event) {
34                 turtlenet.stopTN(new AsyncCallback<String>() {
35                     public void onFailure(Throwable caught) {
36                         //pretend nothing happened
37                     }
38                     public void onSuccess(String result) {
39                         //bask in success
40                     }
41                 });
42             }
43         });
44
45         // Call method to load the initial login page
46         login();
47     }
48
49     private String location = new String("");
50     private String refreshID = new String("");
51
52     // LOUISTODO May need to remove ' = new FlexTable()'
53     private FlexTable loginPanel = new FlexTable();
54     private void login() {
55         location = "login";
56         refreshID = "";
57         RootPanel.get().clear();
58         loginPanel = new FlexTable();
59         loginPanel.clear();
60         RootPanel.get().add(loginPanel);
61
62         // Create login panel widgets
63         final Button loginButton = new Button("Login");
64         loginButton.addClickListener(this);
65         final PasswordTextBox passwordInput = new PasswordTextBox();
66         final Label passwordLabel = new Label();
67
68         turtlenet.isFirstTime(new AsyncCallback<String>() {
69             public void onFailure(Throwable caught) {
70                 System.out.println("turtlenet.isFirstTime failed: " + caught);
71             }
72             public void onSuccess(String result) {
73                 if(result.equals("true")) { //GWT can only return objects
74                     passwordLabel.setText("Please choose a password:");
75                     final PasswordTextBox passwordConfirmInput = new PasswordTextBox();
76                     final Label passwordConfirmLabel = new Label("");
77                     passwordConfirmLabel.setText("Confirm your password:");
78                     final TextBox usernameInput = new TextBox();
79                     final Label usernameLabel = new Label("");
80                     usernameLabel.setText("Please choose a username:");
81
82                     // Add widgets to login panel
83                     loginPanel.setWidget(1, 1, usernameLabel);
84                     loginPanel.setWidget(2, 1, usernameInput);
85                     loginPanel.setWidget(3, 1, passwordLabel);
86                     loginPanel.setWidget(4, 1, passwordInput);
87                     loginPanel.setWidget(5, 1, passwordConfirmLabel);
88                     loginPanel.setWidget(6, 1, passwordConfirmInput);
89                     loginPanel.setWidget(7, 1, loginButton);
90
91                     // Add click handler for button
92                     loginButton.addClickHandler(new ClickHandler() {
93                         public void onClick(ClickEvent event) {

```

```

94         passwordLabel.setText("Please choose a password:");
95         passwordLabel.getElement().getStyle().setProperty("color", "#000000");
96         passwordConfirmLabel.setText("Confirm your password");
97         passwordConfirmLabel.getElement().getStyle().setProperty("color", "#000000");
98         usernameLabel.setText("Please choose a username:");
99         usernameLabel.getElement().getStyle().setProperty("color", "#000000");
100
101         if(usernameInput.getText().equals("")) {
102             usernameLabel.setText("Must enter a username");
103             usernameLabel.getElement().getStyle().setProperty("color", "#FFFFFF");
104         } else if(passwordInput.getText().equals("")) {
105             passwordLabel.setText("Must enter a password");
106             passwordLabel.getElement().getStyle().setProperty("color", "#FFFFFF");
107         } else if(passwordConfirmInput.getText().equals("")) {
108             passwordConfirmLabel.setText("Must confirm password");
109             passwordConfirmLabel.getElement().getStyle().setProperty("color", "#FFFFFF");
110         } else if(passwordInput.getText().equals(passwordConfirmInput.getText())) {
111             turtlenet.register(usernameInput.getText(), passwordInput.getText(), new
112 AsyncCallback<String>() {
113
114             public void onFailure(Throwable caught) {
115                 System.out.println("turtlenet.register failed: " + caught);
116             }
117             public void onSuccess(String result) {
118                 if (result.equals("success")) {
119                     turtlenet.getMyKey(new AsyncCallback<String>() {
120                         public void onFailure(Throwable caught) {
121                             System.out.println("turtlenet.getMyKey failed: " + caught);
122                         }
123                         public void onSuccess(String result) {
124                             wall(result, false);
125                         }
126                     });
127                 } else if (result.equals("taken")) {
128                     usernameLabel.setText("Username already taken. Try again:");
129                     usernameLabel.getElement().getStyle().setProperty("color", "#FFFFFF");
130                 } else {
131                     System.out.println("turtlenet.register onSuccess String result did not equal
132 success or taken");
133                 }
134             }
135         });
136     } else {
137         passwordLabel.setText("Passwords do not match. Try again:");
138         passwordLabel.getElement().getStyle().setProperty("color", "#FFFFFF");
139         passwordConfirmInput.setText("");
140         passwordInput.setText("");
141     }
142 } else {
143     passwordLabel.setText("Please enter your password:");
144
145     // Add widgets to login panel
146     loginPanel.setWidget(1, 1, passwordLabel);
147     loginPanel.setWidget(2, 1, passwordInput);
148     loginPanel.setWidget(3, 1, loginButton);
149
150     // Add click handler for button
151     loginButton.addClickHandler(new ClickHandler() {
152         public void onClick(ClickEvent event) {
153             passwordLabel.setText("Please enter your password:");
154
155             turtlenet.startTN(passwordInput.getText(), new AsyncCallback<String>() {
156                 public void onFailure(Throwable caught) {
157                     System.out.println("turtlenet.startTN failed: " + caught);
158                 }
159                 public void onSuccess(String result) {
160                     if (result.equals("success")) {
161                         turtlenet.getMyKey(new AsyncCallback<String>() {
162                             public void onFailure(Throwable caught) {
163                                 System.out.println("turtlenet.getMyKey failed: " + caught);
164                             }
165                             public void onSuccess(String result) {
166                                 wall(result, false);
167                             }
168                         });
169                     } else if (result.equals("failure")) {
170                         passwordLabel.setText("Password incorrect. Try again: ");
171                     } else {
172                         System.out.println("turtlenet.startTN onSuccess String does not equal success or
173 failure");
174                         passwordLabel.setText("INVALID RESPONSE FROM TNclient");
175                     }
176                 }
177             });
178         }
179     });
180 }
181 });
182
183 // Add style name for CSS

```

```

184     loginPanel.addStyleName("gwt-login");
185 }
186
187 // Used to track the most recent wall post to be displayed
188 Long wallLastTimeStamp = 0L;
189 Long conversationLastTimeStamp = 0L;
190 Long commentsLastTimeStamp = 0L;
191
192 // When the login button is clicked we start a repeating timer that refreshes
193 // the page every 5 seconds.
194 public void onClick(Widget sender) {
195     Timer refresh = new Timer() {
196         public void run() {
197             if(location.equals("wall")) {
198                 turtlenet.timeMostRecentWallPost(refreshID, new AsyncCallback<Long>() {
199                     public void onFailure(Throwable caught) {
200                         System.out.println("turtlenet.timeMostRecentWallPost failed: " + caught);
201                     }
202                     public void onSuccess(Long result) {
203                         if(result > wallLastTimeStamp) {
204                             System.out.println("Refreshing wall. refreshID: " + refreshID);
205                             wall(refreshID, true);
206                         }
207                     }
208                 });
209             } else if(location.equals("conversationList")) {
210                 System.out.println("Refreshing conversationList");
211                 conversationList();
212             } else if(location.equals("conversation")) {
213                 turtlenet.getConvoLastUpdated(refreshID, new AsyncCallback<Long>() {
214                     public void onFailure(Throwable caught) {
215                         //TODO Error
216                     }
217                     public void onSuccess(Long result) {
218                         if(result > conversationLastTimeStamp) {
219                             System.out.println("Refreshing conversation. refreshID: " + refreshID);
220                             conversation(refreshID, true);
221                         }
222                     }
223                 });
224             } else {
225                 //Do nothing
226             }
227         }
228     };
229     refresh.scheduleRepeating(5*1000);
230 }
231
232 private void navigation() {
233     HorizontalPanel navigationPanel = new HorizontalPanel();
234     RootPanel.get().add(navigationPanel);
235
236     // Create navigation links
237     Anchor linkMyWall = new Anchor("My Wall");
238     linkMyWall.getElement().getStyle().setProperty("paddingLeft", "100px");
239     Anchor linkMyDetails = new Anchor("My Details");
240     linkMyDetails.getElement().getStyle().setProperty("paddingLeft", "100px");
241     Anchor linkConversations = new Anchor("Messages");
242     linkConversations.getElement().getStyle().setProperty("paddingLeft", "100px");
243     Anchor linkFriends = new Anchor("Friends");
244     linkFriends.getElement().getStyle().setProperty("paddingLeft", "100px");
245     Anchor linkLogout = new Anchor("Logout");
246     linkLogout.getElement().getStyle().setProperty("paddingLeft", "100px");
247
248     // Add links to navigation panel
249     navigationPanel.add(linkMyWall);
250     navigationPanel.add(linkMyDetails);
251     navigationPanel.add(linkConversations);
252     navigationPanel.add(linkFriends);
253     navigationPanel.add(linkLogout);
254
255     // Add style name for CSS
256     navigationPanel.addStyleName("gwt-navigation");
257
258     // Add click handlers for anchors
259     linkMyWall.addClickHandler(new ClickHandler() {
260         public void onClick(ClickEvent event) {
261             turtlenet.getMyKey(new AsyncCallback<String>() {
262                 public void onFailure(Throwable caught) {
263                     System.out.println("turtlenet.getMyKey failed: " + caught);
264                 }
265                 public void onSuccess(String result) {
266                     wall(result, false);
267                 }
268             });
269         }
270     });
271
272     // Add click handlers for anchors
273     linkMyDetails.addClickHandler(new ClickHandler() {
274         public void onClick(ClickEvent event) {
275             myDetails();
276         }
277     });

```

```

277     });
278
279     linkConversations.addClickHandler(new ClickHandler() {
280         public void onClick(ClickEvent event) {
281             conversationList();
282
283             System.out.println("Wake up, Neo...");
284         }
285     });
286
287     linkFriends.addClickHandler(new ClickHandler() {
288         public void onClick(ClickEvent event) {
289             friendsList("All");
290         }
291     });
292
293     linkLogout.addClickHandler(new ClickHandler() {
294         public void onClick(ClickEvent event) {
295             turtlenet.stopTN(new AsyncCallback<String>() {
296                 public void onFailure(Throwable caught) {
297                     System.out.println("turtlenet.stopTN failed: " + caught);
298                 }
299                 public void onSuccess(String result) {
300                     login();
301                 }
302             });
303         }
304     });
305
306     String[][] friendsListCategoryMembers = new String[0][0];
307     String[][] friendsListCategoryList = new String[0][0];
308     private TextBox friendsListPanel_myKeyTextBox;
309     private void friendsList(final String currentGroupID) {
310         location = "friendsList";
311         refreshID = "";
312
313         RootPanel.get().clear();
314         navigation();
315         final FlexTable friendsListPanel = new FlexTable();
316         RootPanel.get().add(friendsListPanel);
317
318         // Column title for anchors linking to messages
319         Label friendsNameLabel = new Label("Friend's Name");
320         friendsNameLabel.getElement().getStyle().setFontWeight(FontWeight.BOLD);
321         friendsNameLabel.getElement().getStyle().setProperty("paddingLeft", "100px");
322         friendsListPanel.setWidget(1, 0, friendsNameLabel);
323
324         // Column title for labels outputting the date a message was recieved
325         Label friendsKeyLabel = new Label("Friend's Public Key");
326         friendsKeyLabel.getElement().getStyle().setFontWeight(FontWeight.BOLD);
327         friendsListPanel.setWidget(1, 1, friendsKeyLabel);
328
329         turtlenet.getCategoryMembers(currentGroupID, new AsyncCallback<String[][]>() {
330             int i;
331             public void onFailure(Throwable caught) {
332                 System.out.println("turtlenet.getCategoryMembers failed: " + caught);
333             }
334             public void onSuccess(String[][] _result) {
335                 friendsListCategoryMembers = _result;
336                 for (i = 0; i < friendsListCategoryMembers.length; i++) {
337                     //list names/keys
338                     Anchor linkFriendsWall = new Anchor(friendsListCategoryMembers[i][0]);
339                     linkFriendsWall.getElement().getStyle().setProperty("paddingLeft", "100px");
340                     friendsListPanel.setWidget((i + 2), 0, linkFriendsWall);
341                     final String resultString = friendsListCategoryMembers[i][1];
342                     TextBox friendKeyBox = new TextBox();
343                     friendKeyBox.setText(resultString);
344                     friendKeyBox.setVisibleLength(75);
345                     friendKeyBox.setReadOnly(true);
346                     friendsListPanel.setWidget((i + 2), 1, friendKeyBox);
347                     //link names to walls
348                     System.out.println("adding link to " + friendsListCategoryMembers[i][0] + "'s wall");
349                     final String fkey = friendsListCategoryMembers[i][1];
350                     linkFriendsWall.addClickHandler(new ClickHandler() {
351                         public void onClick(ClickEvent event) {
352                             wall(fkey, false);
353                         }
354                     });
355                 }
356             }
357         });
358
359         int row = friendsListPanel.getRowCount() + 2;
360
361         if(!currentGroupID.equals("All")) {
362             Label currentGroupLabel = new Label(currentGroupID);
363             friendsListPanel.setWidget((row - 1), 3, currentGroupLabel);
364         }
365
366         final ListBox currentGroups = new ListBox();
367         currentGroups.setVisibleItemCount(1);
368         currentGroups.setWidth("150px");

```

```

369     currentGroups.addItem("All");
370     friendsListPanel.setWidget(3, 3, currentGroups);
371
372     turtlenet.getCategories(new AsyncCallback<String[][]>() {
373         int i;
374         public void onFailure(Throwable caught) {
375             System.out.println("turtlenet.getCategories failed: " + caught);
376         }
377         int selected;
378         public void onSuccess(String[][] _result) {
379             friendsListCategoryList = _result;
380             for (i = 0; i < friendsListCategoryList.length; i++) {
381                 currentGroups.addItem(friendsListCategoryList[i][0]);
382                 // Check if the group we've just added is the current group
383                 // If it is note the index using selected. We need to add
384                 // 1 to selected as "All" always appears first in the list.
385                 if(friendsListCategoryList[i][0].equals(currentGroupID)) {
386                     selected = (i + 1);
387                 }
388             }
389             // Use selected to set the selected item in the listbox to the
390             // current group
391             currentGroups.setSelectedIndex(selected);
392
393             currentGroups.addChangeHandler(new ChangeHandler() {
394                 public void onChange(ChangeEvent event) {
395                     friendsList(currentGroups.getItemText(currentGroups.getSelectedIndex()));
396                 }
397             });
398         }
399     });
400
401     Button newGroup = new Button("Add new category");
402     friendsListPanel.setWidget(2, 3, newGroup);
403     newGroup.addClickHandler(new ClickHandler() {
404         public void onClick(ClickEvent event) {
405             newGroup();
406         }
407     });
408
409     friendsListPanel_myKeyTextBox = new TextBox();
410     friendsListPanel_myKeyTextBox.setWidth("480px");
411     friendsListPanel_myKeyTextBox.setReadOnly(true);
412
413     turtlenet.getMyKey(new AsyncCallback<String>() {
414         public void onFailure(Throwable caught) {
415             System.out.println("turtlenet.getMyKey failed: " + caught);
416         }
417         public void onSuccess(String result) {
418             friendsListPanel_myKeyTextBox.setText(result);
419         }
420     });
421
422     Label myKeyLabel = new Label("My key: ");
423     myKeyLabel.getElement().getStyle().setFontWeight(FontWeight.BOLD);
424     myKeyLabel.getElement().getStyle().setProperty("paddingLeft", "100px");
425     friendsListPanel.setWidget((row - 1), 0, myKeyLabel);
426     friendsListPanel.setWidget((row - 1), 1, friendsListPanel_myKeyTextBox);
427
428     if(currentGroupID.equals("All")) {
429         Button addFriend = new Button("Add new friend");
430         friendsListPanel.setWidget(1, 3, addFriend);
431         addFriend.addClickHandler(new ClickHandler() {
432             public void onClick(ClickEvent event) {
433                 addFriend();
434             }
435         });
436     } else {
437         Button editGroup = new Button("Edit category");
438         friendsListPanel.setWidget(1, 3, editGroup);
439         editGroup.addClickHandler(new ClickHandler() {
440             public void onClick(ClickEvent event) {
441                 editGroup(currentGroupID);
442             }
443         });
444     }
445
446     // Add style name for CSS
447     friendsListPanel.addStyleName("gwt-friends-list");
448 }
449
450 FlexTable conversationListPanel;
451 private void conversationList() {
452     location = "conversationList";
453     refreshID = "";
454
455     //Setup basic page
456     RootPanel.get().clear();
457     navigation();
458
459     //Create panel to contain widgets
460     conversationListPanel = new FlexTable();
461     RootPanel.get().add(conversationListPanel);

```

```

462
463     turtlenet.getConversations(new AsyncCallback<Conversation[]>() {
464         Conversation[] result;
465         public void onFailure(Throwable caught) {
466             System.out.println("turtlenet.getConversations failed: " + caught);
467         }
468         public void onSuccess(Conversation[] _result) {
469             result = _result;
470             for(int j = 0; j < result.length; j++) {
471                 System.out.println(result[j]);
472             }
473             System.out.println("result.length = " + result.length);
474             for (int i = 0; i < result.length; i++) {
475                 final String conversationID = result[i].signature;
476                 // Substrings dont work if we set the end point so its
477                 // bigger than our string. If the length is less than 40
478                 // we output the full string. If the string is 40 or
479                 // about we take the first 40 characters and add ...
480                 String linkText = new String("");
481                 if ((result[i].firstMessage).length() < 40) {
482                     linkText = (result[i].firstMessage);
483                 } else {
484                     linkText = (result[i].firstMessage).substring(1, 40) + "...";
485                 }
486                 Anchor linkConversation = new Anchor(linkText);
487                 conversationListPanel.setWidget(i, 0, linkConversation);
488
489                 // Add click handlers for anchors
490                 linkConversation.addClickHandler(new ClickHandler() {
491                     public void onClick(ClickEvent event) {
492                         conversation(conversationID, false);
493                     }
494                 });
495                 Label conversationParticipants = new Label(result[i].concatNames());
496                 conversationListPanel.setWidget(i, 1, conversationParticipants);
497             }
498             Button newConversation = new Button("New conversation");
499             newConversation.setWidth("400px");
500             newConversation.addClickHandler(new ClickHandler() {
501                 public void onClick(ClickEvent event) {
502                     newConversation();
503                 }
504             });
505
506             conversationListPanel.setWidget((result.length + 1), 0, newConversation);
507         }
508     });
509
510     // Add style name for CSS
511     conversationListPanel.addStyleName("gwt-conversation-list");
512 }
513
514 private void myDetails() {
515     location = "myDetails";
516     refreshID = "";
517
518     RootPanel.get().clear();
519     navigation();
520     FlexTable myDetailsPanel = new FlexTable();
521     RootPanel.get().add(myDetailsPanel);
522
523     // Create widgets relating to username
524     Label usernameLabel = new Label("Username:");
525     myDetailsPanel.setWidget(0, 0, usernameLabel);
526
527     final TextBox editUsername = new TextBox();
528     editUsername.setWidth("300px");
529     turtlenet.getMyUsername(new AsyncCallback<String>() {
530         public void onFailure(Throwable caught) {
531             System.out.println("turtlenet.getMyUsername failed: " + caught);
532         }
533         public void onSuccess(String result) {
534             editUsername.setText(result);
535         }
536     });
537
538     myDetailsPanel.setWidget(0, 1, editUsername);
539
540     Button saveUsername = new Button("Save Username");
541     myDetailsPanel.setWidget(0, 2, saveUsername);
542
543     final Label editUsernameLabel = new Label();
544     editUsernameLabel.setWidth("200px");
545     myDetailsPanel.setWidget(0, 3, editUsernameLabel);
546
547     saveUsername.addClickHandler(new ClickHandler() {
548         public void onClick(ClickEvent event) {
549             turtlenet.claimUsername(editUsername.getText(), new AsyncCallback<String>() {
550                 public void onFailure(Throwable caught) {
551                     System.out.println("turtlenet.claimUsername failed: " + caught);
552                 }
553                 public void onSuccess(String result) {
554                     if (result.equals("success")) {

```



```

555         editUsernameLabel.setText("Username saved");
556     } else if (result.equals("failure")) {
557         editUsernameLabel.setText("Username taken");
558     }
559     }
560     });
561 }
562 });
563
564 // Create widgets relating to name
565 Label nameLabel = new Label("Name:");
566 myDetailsPanel.setWidget(1, 0, nameLabel);
567
568 final TextBox editName = new TextBox();
569 editName.setWidth("300px");
570 turtlenet.getMyPDATA("name", new AsyncCallback<String>() {
571     public void onFailure(Throwable caught) {
572         System.out.println("turtlenet.getMyPDATA name failed: " + caught);
573     }
574     public void onSuccess(String result) {
575         editName.setText(result);
576     }
577 });
578 myDetailsPanel.setWidget(1, 1, editName);
579
580 Button saveName = new Button("Save Name");
581 myDetailsPanel.setWidget(1, 2, saveName);
582
583 final Label editNameLabel = new Label();
584 myDetailsPanel.setWidget(1, 3, editNameLabel);
585
586 saveName.addClickHandler(new ClickHandler() {
587     public void onClick(ClickEvent event) {
588         turtlenet.updatePDATA("name", editName.getText(), new AsyncCallback<String>() {
589             public void onFailure(Throwable caught) {
590                 System.out.println("turtlenet.updatePDATA name failed: " + caught);
591             }
592             public void onSuccess(String result) {
593                 if (result.equals("success")) {
594                     editNameLabel.setText("Name saved");
595                 } else if (result.equals("failure")) {
596                     // HORRIBLE FIX
597                     //editNameLabel.setText("Failed to save name");
598                     editNameLabel.setText("Name saved");
599                 }
600             }
601         });
602     }
603 });
604
605 // Create widgets relating to birthday
606 Label birthdayLabel = new Label("Birthday:");
607 myDetailsPanel.setWidget(2, 0, birthdayLabel);
608
609 final TextBox editBirthday = new TextBox();
610 editBirthday.setWidth("300px");
611 turtlenet.getMyPDATA("birthday", new AsyncCallback<String>() {
612     public void onFailure(Throwable caught) {
613         System.out.println("turtlenet.getMyPDATA birthday failed: " + caught);
614     }
615     public void onSuccess(String result) {
616         editBirthday.setText(result);
617     }
618 });
619 myDetailsPanel.setWidget(2, 1, editBirthday);
620
621 Button saveBirthday = new Button("Save Birthday");
622 myDetailsPanel.setWidget(2, 2, saveBirthday);
623
624 final Label editBirthdayLabel = new Label();
625 myDetailsPanel.setWidget(2, 3, editBirthdayLabel);
626
627 saveBirthday.addClickHandler(new ClickHandler() {
628     public void onClick(ClickEvent event) {
629         turtlenet.updatePDATA("birthday", editBirthday.getText(), new AsyncCallback<String>() {
630             public void onFailure(Throwable caught) {
631                 System.out.println("turtlenet.updatePDATA birthday failed: " + caught);
632             }
633             public void onSuccess(String result) {
634                 if (result.equals("success")) {
635                     editBirthdayLabel.setText("Birthday saved");
636                 } else if (result.equals("failure")) {
637                     // HORRIBLE FIX
638                     //editBirthdayLabel.setText("Failed to save birthday");
639                     editBirthdayLabel.setText("Birthday saved");
640                 }
641             }
642         });
643     }
644 });
645
646 // Create widgets relating to gender
647 Label genderLabel = new Label("Gender:");

```

```

648     myDetailsPanel.setWidget(3, 0, genderLabel);
649
650     final TextBox editGender = new TextBox();
651     editGender.setWidth("300px");
652     turtlenet.getMyPDATA("gender", new AsyncCallback<String>() {
653         public void onFailure(Throwable caught) {
654             System.out.println("turtlenet.getMyPDATA gender failed: " + caught);
655         }
656         public void onSuccess(String result) {
657             editGender.setText(result);
658         }
659     });
660     myDetailsPanel.setWidget(3, 1, editGender);
661
662     Button saveGender = new Button("Save Gender");
663     myDetailsPanel.setWidget(3, 2, saveGender);
664
665     final Label editGenderLabel = new Label();
666     myDetailsPanel.setWidget(3, 3, editGenderLabel);
667
668     saveGender.addClickHandler(new ClickHandler() {
669         public void onClick(ClickEvent event) {
670             turtlenet.updatePDATA("gender", editGender.getText(), new AsyncCallback<String>() {
671                 public void onFailure(Throwable caught) {
672                     System.out.println("turtlenet.updatePDATA gender failed: " + caught);
673                 }
674                 public void onSuccess(String result) {
675                     if (result.equals("success")) {
676                         editGenderLabel.setText("Gender saved");
677                     } else if (result.equals("failure")) {
678                         // HORRIBLE FIX
679                         //editGenderLabel.setText("Failed to save gender");
680                         editGenderLabel.setText("Gender saved");
681                     }
682                 }
683             });
684         }
685     });
686
687     // Create widgets relating to email
688     final Label emailLabel = new Label("Email:");
689     myDetailsPanel.setWidget(4, 0, emailLabel);
690
691     final TextBox editEmail = new TextBox();
692     editEmail.setWidth("300px");
693     turtlenet.getMyPDATA("email", new AsyncCallback<String>() {
694         public void onFailure(Throwable caught) {
695             System.out.println("turtlenet.getMyPDATA email failed: " + caught);
696         }
697         public void onSuccess(String result) {
698             editEmail.setText(result);
699         }
700     });
701     myDetailsPanel.setWidget(4, 1, editEmail);
702
703     Button saveEmail = new Button("Save Email");
704     myDetailsPanel.setWidget(4, 2, saveEmail);
705
706     final Label editEmailLabel = new Label();
707     myDetailsPanel.setWidget(4, 3, editEmailLabel);
708
709     saveEmail.addClickHandler(new ClickHandler() {
710         public void onClick(ClickEvent event) {
711             turtlenet.updatePDATA("email", editEmail.getText(), new AsyncCallback<String>() {
712                 public void onFailure(Throwable caught) {
713                     System.out.println("turtlenet.updatePDATA email failed: " + caught);
714                 }
715                 public void onSuccess(String result) {
716                     if (result.equals("success")) {
717                         editEmailLabel.setText("Email saved");
718                     } else if (result.equals("failure")) {
719                         // HORRIBLE FIX
720                         //editEmailLabel.setText("Failed to save email");
721                         editEmailLabel.setText("Email saved");
722                     }
723                 }
724             });
725         }
726     });
727
728     Button revoke = new Button("Revoke Key");
729     myDetailsPanel.setWidget(5, 1, revoke);
730     revoke.getElement().getStyle().setProperty("color", "#FF0000");
731     revoke.setWidth("310px");
732
733     final Label editkeyRevokeLabel = new Label();
734     myDetailsPanel.setWidget(5, 3, editkeyRevokeLabel);
735
736     revoke.addClickHandler(new ClickHandler() {
737         public void onClick(ClickEvent event) {
738             turtlenet.revokeMyKey(new AsyncCallback<String>() {
739                 public void onFailure(Throwable caught) {
740                     System.out.println("turtlenet.revokeMyKey failed: " + caught);

```



```

741         }
742         public void onSuccess(String result) {
743             //if (result.equals("success")) {
744                 editEmailLabel.setText("Key revoked");
745                 login();
746             //} else if (result.equals("failure")) {
747                 //editEmailLabel.setText("Failed to revoke key");
748             //}
749         }
750     });
751 }
752 });
753
754 myDetailsPermissions();
755
756 // Add style name for CSS
757 myDetailsPanel.addStyleName("gwt-my-details");
758 }
759
760 private void myDetailsPermissions() {
761     location = "myDetailsPermissions";
762     refreshID = "";
763
764     // Add panel to contain widgets
765     final FlexTable myDetailsPermissionsPanel = new FlexTable();
766     RootPanel.get().add(myDetailsPermissionsPanel);
767
768     Label keyRevokeLabel = new Label("If you revoke your key your account will be deleted!");
769     keyRevokeLabel.getElement().getStyle().setProperty("color", "#FF0000");
770     myDetailsPermissionsPanel.setWidget(0, 0, keyRevokeLabel);
771
772     Label myDetailsPermissionsLabel = new Label("Select which groups can view your details:");
773     myDetailsPermissionsLabel.getElement().getStyle().setFontWeight(FontWeight.BOLD);
774     myDetailsPermissionsPanel.setWidget(1, 0, myDetailsPermissionsLabel);
775
776     turtlenet.getCategories(new AsyncCallback<String[][]>() {
777         String[][] result;
778         int i;
779         public void onFailure(Throwable caught) {
780             System.out.println("turtlenet.getCategories failed: " + caught);
781         }
782         public void onSuccess(String[][] _result) {
783             result = _result;
784             for (i = 0; i < result.length; i++) {
785                 final CheckBox groupCheckBox = new CheckBox(result[i][0]);
786                 groupCheckBox.setValue(result[i][1].equals("true"));
787                 myDetailsPermissionsPanel.setWidget((i + 1), 0, groupCheckBox);
788
789                 groupCheckBox.addClickHandler(new ClickHandler() {
790                     public void onClick(ClickEvent event) {
791                         turtlenet.updatePDATAPermission(groupCheckBox.getText(), groupCheckBox.getValue(), new
792 AsyncCallback<String>() {
793                             public void onFailure(Throwable caught) {
794                                 System.out.println("updatePDATAPermission failed: " + caught);
795                             }
796                             public void onSuccess(String result) {
797                                 //success
798                             }
799                         });
800                     }
801                 });
802             }
803         });
804     myDetailsPermissionsPanel.addStyleName("gwt-my-details-permissions");
805 }
806
807 private void friendsDetails(final String friendsDetailsKey, FlowPanel wallPanel, Button userDetails) {
808     userDetails.addClickHandler(new ClickHandler() {
809         public void onClick(ClickEvent event) {
810             wall(friendsDetailsKey, false);
811         }
812     });
813
814     userDetails.setText("Reload page");
815     userDetails.getElement().getStyle().setProperty("color", "#61B329");
816
817     location = "friendsDetails";
818     refreshID = "";
819
820     // Create main panel
821     final FlexTable friendsDetailsPanel = new FlexTable();
822     wallPanel.insert(friendsDetailsPanel, 1);
823     friendsDetailsPanel.clear();
824
825     // Create widgets
826     Label friendsDetailsUsernameTitle = new Label("Username:");
827     friendsDetailsPanel.setWidget(0, 0, friendsDetailsUsernameTitle);
828
829     Label friendsDetailsNameTitle = new Label("Name:");
830     friendsDetailsPanel.setWidget(1, 0, friendsDetailsNameTitle);
831
832     Label friendsDetailsBirthdayTitle = new Label("Birthday:");

```

```

833         friendsDetailsPanel.setWidget(2, 0, friendsDetailsBirthDayTitle);
834
835         Label friendsDetailsGenderTitle = new Label("Gender:");
836         friendsDetailsPanel.setWidget(3, 0, friendsDetailsGenderTitle);
837
838         Label friendsDetailsEmailTitle = new Label("Email:");
839         friendsDetailsPanel.setWidget(4, 0, friendsDetailsEmailTitle);
840
841         Label friendsDetailsKeyTitle = new Label("Public Key:");
842         friendsDetailsPanel.setWidget(5, 0, friendsDetailsKeyTitle);
843
844         turtlenet.getUsername(friendsDetailsKey, new AsyncCallback<String>() {
845             public void onFailure(Throwable caught) {
846                 System.out.println("turtlenet.getUsername failed: " + caught);
847             }
848             public void onSuccess(String result) {
849                 Label friendsDetailsUsernameLabel = new Label(result);
850                 friendsDetailsPanel.setWidget(0, 1, friendsDetailsUsernameLabel);
851             }
852         });
853
854         turtlenet.getPDATA("name", friendsDetailsKey, new AsyncCallback<String>() {
855             public void onFailure(Throwable caught) {
856                 System.out.println("turtlenet.getPDATA name failed: " + caught);
857             }
858             public void onSuccess(String result) {
859                 Label friendsDetailsNameLabel = new Label(result);
860                 friendsDetailsPanel.setWidget(1, 1, friendsDetailsNameLabel);
861             }
862         });
863
864         turtlenet.getPDATA("birthday", friendsDetailsKey, new AsyncCallback<String>() {
865             public void onFailure(Throwable caught) {
866                 System.out.println("turtlenet.getPDATA birthday failed: " + caught);
867             }
868             public void onSuccess(String result) {
869                 Label friendsDetailsBirthDayLabel = new Label(result);
870                 friendsDetailsPanel.setWidget(2, 1, friendsDetailsBirthDayLabel);
871             }
872         });
873
874         turtlenet.getPDATA("gender", friendsDetailsKey, new AsyncCallback<String>() {
875             public void onFailure(Throwable caught) {
876                 System.out.println("turtlenet.getPDATA gender failed: " + caught);
877             }
878             public void onSuccess(String result) {
879                 Label friendsDetailsGenderLabel = new Label(result);
880                 friendsDetailsPanel.setWidget(3, 1, friendsDetailsGenderLabel);
881             }
882         });
883
884         turtlenet.getPDATA("email", friendsDetailsKey, new AsyncCallback<String>() {
885             public void onFailure(Throwable caught) {
886                 System.out.println("turtlenet.getPDATA email failed: " + caught);
887             }
888             public void onSuccess(String result) {
889                 Label friendsDetailsEmailLabel = new Label(result);
890                 friendsDetailsPanel.setWidget(4, 1, friendsDetailsEmailLabel);
891             }
892         });
893
894         TextBox friendsDetailsKeyBox = new TextBox();
895         friendsDetailsKeyBox.setReadOnly(true);
896         friendsDetailsKeyBox.setWidth("400px");
897         friendsDetailsKeyBox.setText(friendsDetailsKey);
898         friendsDetailsPanel.setWidget(5, 1, friendsDetailsKeyBox);
899
900         turtlenet.getMyKey(new AsyncCallback<String>() {
901             public void onFailure(Throwable caught) {
902                 System.out.println("turtlenet.getMyKey failed: " + caught);
903             }
904             public void onSuccess(String myKey) {
905                 if(friendsDetailsKey.equals(myKey)) {
906                     Button edit = new Button("Edit my details");
907                     edit.setWidth("410px");
908                     friendsDetailsPanel.setWidget(6, 1, edit);
909                     edit.addClickHandler(new ClickHandler() {
910                         public void onClick(ClickEvent event) {
911                             myDetails();
912                         }
913                     });
914                 }
915             }
916         });
917
918         // Add style name for CSS
919         friendsDetailsPanel.addStyleName("gwt-friends-details");
920     }
921
922     // Global stuff for wall
923     private HorizontalPanel wallControlPanel = new HorizontalPanel();
924     private TextArea postText;
925     PostDetails[] wallPostDetails;

```

```

926     int wallCurrentPost;
927     private FlowPanel wallPanel = new FlowPanel();
928     private Button wallControlPanelUserDetailsButton;
929     private Anchor linkToComments;
930
931     private void wall(final String key, final boolean refresh) {
932         location = "wall";
933         refreshID = key;
934
935         wallPanel = new FlowPanel();
936         wallPanel.clear();
937
938         if(!refresh) {
939             // Setup basic page
940             RootPanel.get().clear();
941             navigation();
942             RootPanel.get().add(wallPanel);
943             // Create a container for controls
944             wallControlPanel = new HorizontalPanel();
945             wallControlPanel.clear();
946             wallControlPanel.addStyleName("gwt-wall-control");
947             wallControlPanel.setSpacing(5);
948             wallPanel.insert(wallControlPanel, 0);
949
950             wallControlPanelUserDetailsButton = new Button("About");
951             wallControlPanelUserDetailsButton.getElement().getStyle().setProperty("color", "#000000");
952             wallControlPanel.add(wallControlPanelUserDetailsButton);
953             wallControlPanelUserDetailsButton.getElement().getStyle().setProperty("marginRight", "150px");
954             wallControlPanelUserDetailsButton.addClickHandler(new ClickHandler() {
955                 public void onClick(ClickEvent event) {
956                     friendsDetails(key, wallPanel, wallControlPanelUserDetailsButton);
957                 }
958             });
959
960             turtlenet.getMyKey(new AsyncCallback<String>() {
961                 public void onFailure(Throwable caught) {
962                     System.out.println("turtlenet.getMyKey failed: " + caught);
963                 }
964                 public void onSuccess(String result) {
965                     if(key.equals(result)) {
966                         wallControlPanelUserDetailsButton.setText("About Me");
967                     } else {
968                         turtlenet.getUsername(key, new AsyncCallback<String>() {
969                             public void onFailure(Throwable caught) {
970                                 System.out.println("turtlenet.getUsername failed: " + caught);
971                             }
972                             public void onSuccess(String result) {
973                                 wallControlPanelUserDetailsButton.setText("About " + result);
974                             }
975                         });
976                     }
977                 }
978             });
979
980             final Button createPost = new Button("Write a post");
981             createPost.getElement().getStyle().setProperty("color", "#000000");
982             wallControlPanel.add(createPost);
983
984             final FlowPanel createPostPanel = new FlowPanel();
985             createPostPanel.addStyleName("gwt-create-post");
986             postText = new TextArea();
987             postText.setCharacterWidth(80);
988             postText.setVisibleLines(10);
989             createPostPanel.add(postText);
990
991             HorizontalPanel createPostControlPanel = new HorizontalPanel();
992             createPostPanel.add(createPostControlPanel);
993
994             final ListBox chooseGroup = new ListBox();
995             chooseGroup.setVisibleItemCount(1);
996             chooseGroup.setWidth("150px");
997             chooseGroup.addItem("All");
998             createPostControlPanel.add(chooseGroup);
999             createPostControlPanel.setCellWidth(chooseGroup, "217px");
1000
1001             turtlenet.getCategories(new AsyncCallback<String[][]>() {
1002                 public void onFailure(Throwable caught) {
1003                     System.out.println("turtlenet.getCategories failed: " + caught);
1004                 }
1005                 public void onSuccess(String result[][]) {
1006                     for (int i = 0; i < result.length; i++)
1007                         chooseGroup.addItem(result[i][0]);
1008                 }
1009             });
1010
1011             Button cancel = new Button("Cancel");
1012             createPostControlPanel.add(cancel);
1013             createPostControlPanel.setCellWidth(cancel, "217px");
1014             cancel.addClickHandler(new ClickHandler() {
1015                 public void onClick(ClickEvent event) {
1016                     wallPanel.remove(wallControlPanel);
1017                     wallPanel.remove(createPostPanel);
1018 
```

```

1019         wall(key, false);
1020     }
1021 });
1022
1023 Button send = new Button("Send");
1024 send.setWidth("150px");
1025 createPostControlPanel.add(send);
1026 send.addClickHandler(new ClickHandler() {
1027     public void onClick(ClickEvent event) {
1028         turtlenet.addPost(key, chooseGroup.getItemText(chooseGroup.getSelectedIndex()), postText.getText(), new
AsyncCallback<String>() {
1029             public void onFailure(Throwable caught) {
1030                 System.out.println("turtlenet.addPost failed: " + caught);
1031             }
1032             public void onSuccess(String result) {
1033                 //if (result.equals("success")) {
1034                     wallPanel.remove(wallControlPanel);
1035                     wallPanel.remove(createPostPanel);
1036                     wall(key, false);
1037                 //} else {
1038                     //System.out.println("turtlenet.addPost onSuccess String result did not equal success");
1039                 //}
1040             }
1041         });
1042     });
1043 });
1044
1045 createPost.addClickHandler(new ClickHandler() {
1046     public void onClick(ClickEvent event) {
1047         location = "createPost";
1048         refreshID = "";
1049         createPost.setText("Updates paused");
1050         createPost.getElement().getStyle().setProperty("color", "#FF0000");
1051         wallPanel.insert(createPostPanel, 1);
1052     }
1053 });
1054 }
1055
1056 turtlenet.getWallPosts(key, new AsyncCallback<PostDetails[]>() {
1057     public void onFailure(Throwable caught) {
1058         System.out.println("turtlenet.getWallPosts failed: " + caught);
1059     }
1060     public void onSuccess(PostDetails[] result) {
1061         wallPostDetails = result;
1062         for (wallCurrentPost = 0; wallCurrentPost < wallPostDetails.length; wallCurrentPost++) {
1063             final PostDetails details = wallPostDetails[wallCurrentPost];
1064
1065             if(!refresh || wallPostDetails[wallCurrentPost].timestamp > wallLastTimeStamp) {
1066                 final FlowPanel postPanel = new FlowPanel();
1067                 postPanel.clear();
1068                 wallPanel.insert(postPanel, 1);
1069                 postPanel.addStyleName("gwt-post-panel");
1070
1071                 HorizontalPanel postControlPanel = new HorizontalPanel();
1072                 postPanel.add(postControlPanel);
1073
1074                 //Name
1075                 Label postedByLabel = new Label("Posted by: ");
1076                 postControlPanel.add(postedByLabel);
1077                 postControlPanel.setCellWidth(postedByLabel, "110");
1078
1079                 Anchor linkToUser = new Anchor(wallPostDetails[wallCurrentPost].posterUsername);
1080                 postControlPanel.add(linkToUser);
1081                 postControlPanel.setCellWidth(linkToUser, "200");
1082                 linkToUser.addClickHandler(new ClickHandler() {
1083                     public void onClick(ClickEvent event) {
1084                         wall(wallPostDetails[wallCurrentPost].posterKey, false);
1085                     }
1086                 });
1087
1088                 //Date
1089                 wallLastTimeStamp = wallPostDetails[wallCurrentPost].timestamp;
1090                 Label dateLabel = new Label(new Date(wallPostDetails[wallCurrentPost].timestamp).toString());
1091                 postControlPanel.add(dateLabel);
1092
1093                 FlowPanel postContentsPanel = new FlowPanel();
1094                 postPanel.clear();
1095                 postPanel.add(postContentsPanel);
1096
1097                 TextArea postContents = new TextArea();
1098                 postContents.setCharacterWidth(80);
1099                 postContents.setVisibleLines(5);
1100                 postContents.setReadOnly(true);
1101
1102                 //Text
1103                 postContents.setText(wallPostDetails[wallCurrentPost].text);
1104                 postContentsPanel.add(postContents);
1105
1106                 final HorizontalPanel postContentsFooterPanel = new HorizontalPanel();
1107                 postContentsFooterPanel.addStyleName("gwt-post-content-footer");
1108                 postContentsPanel.add(postContentsFooterPanel);
1109
1110                 //Like

```

```

1111         Anchor likePost;
1112
1113         if (wallPostDetails[wallCurrentPost].liked) {
1114             likePost = new Anchor("Unlike");
1115             likePost.addClickHandler(new ClickHandler() {
1116                 public void onClick(ClickEvent event) {
1117                     turtlenet.unlike(details.sig, new AsyncCallback<String>() {
1118                         public void onFailure(Throwable caught) {
1119                             System.out.println("turtlenet.unlike (post) failed: " + caught);
1120                         }
1121                         public void onSuccess(String _result) {
1122                             //if (_result.equals("success")) {
1123                                 wall(key, false);
1124                             //} else {
1125                                 //System.out.println("turtlenet.unlike (post) onSuccess String _result did
1126                             not equal success");
1127                             //}
1128                         }
1129                     });
1130                 }
1131             } else {
1132                 likePost = new Anchor("Like");
1133                 likePost.addClickHandler(new ClickHandler() {
1134                     public void onClick(ClickEvent event) {
1135                         turtlenet.like(details.sig, new AsyncCallback<String>() {
1136                             public void onFailure(Throwable caught) {
1137                                 System.out.println("turtlenet.like (post) failed: " + caught);
1138                             }
1139                             public void onSuccess(String _result) {
1140                                 //if (_result.equals("success")) {
1141                                     wall(key, false);
1142                                 //} else {
1143                                     //System.out.println("turtlenet.like (post) onSuccess String _result did not
1144                                 equal success");
1145                                 //}
1146                             }
1147                         });
1148                     }
1149                 });
1150             }
1151             postContentsFooterPanel.add(likePost);
1152             final Label stop = new Label("");
1153
1154             //Comments
1155             int commentCount = wallPostDetails[wallCurrentPost].commentCount;
1156             if (commentCount == 0) {
1157                 linkToComments = new Anchor("Add a comment");
1158             } else {
1159                 linkToComments = new Anchor("Comments(" + Integer.toString(commentCount) + ")");
1160             }
1161             linkToComments.getElement().getStyle().setProperty("paddingRight", "100px");
1162             postContentsFooterPanel.add(linkToComments);
1163             linkToComments.addClickHandler(new ClickHandler() {
1164                 public void onClick(ClickEvent event) {
1165                     postContentsFooterPanel.remove(linkToComments);
1166                     stop.setText("Page auto update paused");
1167                     stop.getElement().getStyle().setProperty("color", "#FF0000");
1168                     comments(details.sig, key, false, postPanel);
1169                 }
1170             });
1171             postContentsFooterPanel.add(stop);
1172             postContentsFooterPanel.add(likePost);
1173             likePost.getElement().getStyle().setProperty("paddingLeft", "270px");
1174         }
1175     }
1176
1177     if (refresh) {
1178         // TODO LOUISTODO use this
1179         //Window.scrollTo(0, (Window.getScrollTop() + 200));
1180     }
1181 }
1182
1183 });
1184
1185 // Add style name for CSS
1186 wallPanel.addStyleName("gwt-wall");
1187 }
1188
1189 // Global stuff for comments
1190 private int commentCount;
1191 private TextArea threadReplyContents;
1192 private String keyOfWallCommentsAreOn = new String("");
1193
1194 private void comments(final String postID, final String wallKey, final boolean refresh, FlowPanel postPanel) {
1195     final FlowPanel commentsPanel = new FlowPanel();
1196     location = "comments";
1197     refreshID = postID;
1198     keyOfWallCommentsAreOn = wallKey;
1199
1200     if (!refresh) {
1201         commentsPanel.clear();

```

```

1202
1203 // Disables the comment anchor for the current post to prevent duplicate
1204 // comment panels being created.
1205 linkToComments.addClickHandler(new ClickHandler() {
1206     public void onClick(ClickEvent event) {
1207         commentsPanel.clear();
1208     }
1209 });
1210
1211 // Add main panel to page
1212 postPanel.add(commentsPanel);
1213 FlexTable commentsReplyThreadPanel = new FlexTable();
1214 commentsReplyThreadPanel.getElement().getStyle().setProperty("paddingLeft", "60px");
1215 commentsPanel.add(commentsReplyThreadPanel);
1216
1217 threadReplyContents = new TextArea();
1218 threadReplyContents.setCharacterWidth(60);
1219 threadReplyContents.setVisibleLines(6);
1220 commentsReplyThreadPanel.setWidget(0, 0, threadReplyContents);
1221
1222 Button cancel = new Button("Cancel");
1223 cancel.setWidth("450px");
1224 commentsReplyThreadPanel.setWidget(1, 0, cancel);
1225 cancel.addClickHandler(new ClickHandler() {
1226     public void onClick(ClickEvent event) {
1227         wall(wallKey, false);
1228     }
1229 });
1230
1231 Button replyToThread;
1232 if(commentCount == 0) {
1233     replyToThread = new Button("Post comment");
1234 } else {
1235     replyToThread = new Button("Reply to thread");
1236 }
1237 replyToThread.setWidth("450px");
1238 commentsReplyThreadPanel.setWidget(2, 0, replyToThread);
1239
1240 replyToThread.addClickHandler(new ClickHandler() {
1241     public void onClick(ClickEvent event) {
1242         turtlenet.addComment(postID, threadReplyContents.getText(), new AsyncCallback<String>() {
1243             public void onFailure(Throwable caught) {
1244                 System.out.println("turtlenet.addComment failed: " + caught);
1245             }
1246             public void onSuccess(String result) {
1247                 //if (result.equals("success")) {
1248                     wall(wallKey, false);
1249                 //} else {
1250                     //System.out.println("turtlenet.addComment onSuccess String result did not equal success");
1251                 //}
1252             }
1253         });
1254     }
1255 });
1256 }
1257
1258 turtlenet.getComments(postID, new AsyncCallback<CommentDetails[]>() {
1259     public void onFailure(Throwable caught) {
1260         System.out.println("turtlenet.getComments failed: " + caught);
1261     }
1262     public void onSuccess(CommentDetails[] result) {
1263         commentCount = result.length;
1264         for (int i = 0; i < result.length; i++) {
1265             if(!refresh || result[i].timestamp > commentsLastTimeStamp) {
1266                 final CommentDetails details = result[i];
1267                 // Create panel to contain the main contents of each comment
1268                 FlowPanel commentsContentsPanel = new FlowPanel();
1269                 commentsContentsPanel.addStyleName("gwt-comments-contents");
1270                 commentsPanel.insert(commentsContentsPanel, commentsPanel.getWidgetCount() - 1);
1271
1272                 final String commentID = result[i].sig;
1273                 // Create widgets
1274                 TextArea commentContents = new TextArea();
1275                 commentContents.setCharacterWidth(60);
1276                 commentContents.setVisibleLines(3);
1277                 commentContents.setReadOnly(true);
1278
1279                 //Text
1280                 commentContents.setText(result[i].text);
1281                 commentsContentsPanel.add(commentContents);
1282
1283                 //Create panel to contain controls for each comment
1284                 HorizontalPanel commentsControlPanel = new HorizontalPanel();
1285                 commentsContentsPanel.add(commentsControlPanel);
1286
1287                 final String postedByKey = result[i].posterKey;
1288
1289                 Label commentPostedByLabel = new Label("Posted by: ");
1290                 commentPostedByLabel.getElement().getStyle().setProperty("paddingLeft", "10px");
1291                 commentsControlPanel.add(commentPostedByLabel);
1292
1293                 Anchor postedBy = new Anchor(result[i].posterName);
1294                 postedBy.getElement().getStyle().setProperty("paddingLeft", "10px");

```



```

1295         commentsControlPanel.add(postedBy);
1296
1297         postedBy.addClickHandler(new ClickHandler() {
1298             public void onClick(ClickEvent event) {
1299                 wall(details.posterKey, false);
1300             }
1301         });
1302
1303         Anchor likeComment;
1304
1305         if (result[i].liked) {
1306             likeComment = new Anchor("Unlike");
1307             likeComment.addClickHandler(new ClickHandler() {
1308                 public void onClick(ClickEvent event) {
1309                     turtlenet.unlike(details.sig, new AsyncCallback<String>() {
1310                         public void onFailure(Throwable caught) {
1311                             System.out.println("turtlenet.unlike (comment) failed: " + caught);
1312                         }
1313                         public void onSuccess(String _result) {
1314                             //if (_result.equals("success")) {
1315                                 wall(wallKey, false);
1316                             //} else {
1317                                 //System.out.println("turtlenet.unlike (comment) onSuccess String _result
1318                                 did not equal success");
1319                             //}
1320                         }
1321                     });
1322                 }
1323             });
1324         } else {
1325             likeComment = new Anchor("Like");
1326             likeComment.addClickHandler(new ClickHandler() {
1327                 public void onClick(ClickEvent event) {
1328                     turtlenet.like(details.sig, new AsyncCallback<String>() {
1329                         public void onFailure(Throwable caught) {
1330                             System.out.println("turtlenet.like (comment) failed: " + caught);
1331                         }
1332                         public void onSuccess(String _result) {
1333                             //if (_result.equals("success")) {
1334                                 wall(wallKey, false);
1335                             //} else {
1336                                 //System.out.println("turtlenet.like (comment) onSuccess String _result did
1337                                 not equal success");
1338                             //}
1339                         }
1340                     });
1341                 }
1342             });
1343         }
1344         likeComment.getElement().getStyle().setProperty("paddingLeft", "200px");
1345         commentsControlPanel.add(likeComment);
1346     }
1347 }
1348 });
1349 commentsPanel.addStyleName("gwt-comments");
1350 }
1351
1352 //must be global because it must be referenced from callback
1353 private TextArea newConvoInput = new TextArea();
1354 private void newConversation() {
1355     location = "newConversation";
1356     refreshID = "";
1357
1358     // Setup basic page
1359     RootPanel.get().clear();
1360     navigation();
1361
1362     // Create panel to contain widgets
1363     final FlexTable newConversationPanel = new FlexTable();
1364     RootPanel.get().add(newConversationPanel);
1365
1366     final ListBox currentFriends = new ListBox();
1367     currentFriends.setVisibleItemCount(11);
1368     currentFriends.setWidth("150px");
1369     newConversationPanel.setWidget(0, 0, currentFriends);
1370
1371     newConvoInput.setCharacterWidth(80);
1372     newConvoInput.setVisibleLines(10);
1373     newConversationPanel.setWidget(0, 1, newConvoInput);
1374
1375     final ListBox chooseFriend = new ListBox();
1376     chooseFriend.setWidth("150px");
1377
1378     turtlenet.getPeople(new AsyncCallback<String[][]>() {
1379         String[][] result;
1380         String[] memberKeys;
1381         int i;
1382         public void onFailure(Throwable caught) {
1383             System.out.println("turtlenet.getPeople failed: " + caught);
1384         }
1385         public void onSuccess(String[][] _result) {

```

```

1386         result = _result;
1387         for (i = 0; i < result.length; i++) {
1388             //fill combo box
1389             chooseFriend.addItem(result[i][0]);
1390             String friendKey = (result[i][1]);
1391             chooseFriend.setValue(i, friendKey);
1392         }
1393         chooseFriend.setVisibleItemCount(1);
1394
1395         FlexTable subPanel = new FlexTable();
1396         newConversationPanel.setWidget(1, 1, subPanel);
1397         subPanel.setWidget(1, 0, new Label("Choose a friend: "));
1398         subPanel.setWidget(1, 1, chooseFriend);
1399
1400         Button addFriend = new Button("Add to the conversation");
1401         subPanel.setWidget(1, 2, addFriend);
1402         addFriend.addClickHandler(new ClickHandler() {
1403             public void onClick(ClickEvent event) {
1404                 currentFriends.addItem(chooseFriend.getSelectedItem());
1405                 currentFriends.setValue((currentFriends.getItemCount() - 1), chooseFriend.getValue
(chOOSEFriend.getSelectedItem()));
1406             }
1407         });
1408
1409         Button send = new Button("Send");
1410         newConversationPanel.setWidget(0, 2, send);
1411
1412         send.addClickHandler(new ClickHandler() {
1413             String[] createChatReturn;
1414             public void onClick(ClickEvent event) {
1415                 memberKeys = new String[currentFriends.getItemCount()+1];
1416                 for (int i = 0; i < currentFriends.getItemCount(); i++) {
1417                     memberKeys[i] = currentFriends.getValue(i);
1418                 }
1419
1420                 turtlenet.getMyKey(new AsyncCallback<String>() {
1421
1422                     public void onFailure(Throwable caught) {
1423                         System.out.println("turtlenet.getMyKey failed: " + caught);
1424                     }
1425                     public void onSuccess(String userkey) {
1426                         memberKeys[memberKeys.length-1] = userkey;
1427                         turtlenet.createCHAT(memberKeys, new AsyncCallback<String[]>() {
1428                             int i;
1429                             public void onFailure(Throwable caught) {
1430                                 System.out.println("createCHAT failed: " + caught);
1431                             }
1432                             public void onSuccess(String[] _ret) {
1433                                 createChatReturn = _ret;
1434                                 if (createChatReturn[0].equals("success")) {
1435                                     turtlenet.addMessageToCHAT(newConvoInput.getText(), createChatReturn[1], new
AsyncCallback<String>() {
1436
1437                                         public void onFailure(Throwable caught) {
1438                                             System.out.println("turtlenet.addMessageToCHAT failed: " + caught);
1439                                         }
1440                                         public void onSuccess(String success) {
1441                                             //if (success.equals("success")) {
1442                                                 conversation(createChatReturn[1], false);
1443                                             //} else {
1444                                                 //System.out.println("turtlenet.addMessageToCHAT onSuccess String
success did not equal success");
1445                                             //}
1446                                         }
1447                                     });
1448                                 } else {
1449                                     // THIS IS TEMPORARY!
1450                                     conversation(createChatReturn[1], false);
1451                                     //System.out.println("turtlenet.createCHAT onSuccess String createChatReturn[0]
did not equal success");
1452                                 }
1453                             }
1454                         });
1455                     }
1456                 });
1457             }
1458         });
1459     });
1460
1461     // Add style name for CSS
1462     newConversationPanel.addStyleName("gwt-conversation");
1463 }
1464
1465 // Global stuff for conversation
1466 private String convoPanelSetup_convoSig; //needed in inner class
1467 private TextArea convoPanelSetup_input = new TextArea();
1468 private FlowPanel conversationPanel;
1469
1470 private void conversation(final String conversationID, final boolean refresh) {
1471     location = "conversation";
1472     refreshID = conversationID;
1473
1474     conversationPanel = new FlowPanel();

```

```

1475         final List<Box> currentFriends = new List<Box>();
1476
1477     if(!refresh) {
1478         conversationPanel.clear();
1479         // Set up basic page
1480         RootPanel.get().clear();
1481         navigation();
1482         RootPanel.get().add(conversationPanel);
1483         HorizontalPanel conversationParticipantsPanel = new HorizontalPanel();
1484         conversationParticipantsPanel.setSpacing(5);
1485         conversationPanel.add(conversationParticipantsPanel);
1486         convoPanelSetup_convoSig = conversationID;
1487         Label participantsLabel = new Label("Participants: ");
1488         participantsLabel.getElement().getStyle().setProperty("marginRight", "20px");
1489         conversationParticipantsPanel.add(participantsLabel);
1490
1491         currentFriends.setVisibleItemCount(1);
1492         currentFriends.setWidth("150px");
1493         conversationParticipantsPanel.add(currentFriends);
1494     }
1495
1496     turtlenet.getConversation(convoPanelSetup_convoSig, new AsyncCallback<Conversation>() {
1497         Conversation result;
1498         int i;
1499         public void onFailure(Throwable caught) {
1500             System.out.println("turtlenet.getConversation failed: " + caught);
1501         }
1502         public void onSuccess(Conversation _result) {
1503             result = _result;
1504
1505             if (!refresh) {
1506                 for (i = 0; i < result.users.length; i++) {
1507                     currentFriends.addItem(result.users[i]);
1508                 }
1509             }
1510
1511             turtlenet.getConversationMessages(convoPanelSetup_convoSig, new AsyncCallback<String[][]>() {
1512                 String[][] messages;
1513                 int i;
1514                 public void onFailure(Throwable caught) {
1515                     System.out.println("turtlenet.getConversationMessages failed: " + caught);
1516                 }
1517                 public void onSuccess(String[][] msgs) {
1518                     messages = msgs;
1519
1520                     Button replyToConversation = new Button("Reply");
1521                     replyToConversation.setWidth("590px");
1522
1523                     for (int i = 0; i < messages.length; i++) {
1524                         if(!refresh || Long.parseLong(msgs[i][1]) > conversationLastTimeStamp) {
1525                             HorizontalPanel conversationContentsPanel = new HorizontalPanel();
1526                             conversationContentsPanel.setSpacing(5);
1527                             conversationPanel.add(conversationContentsPanel);
1528                             Label postedBy = new Label(messages[i][0]);
1529                             postedBy.getElement().getStyle().setProperty("marginRight", "110px");
1530                             postedBy.getElement().getStyle().setFontWeight(FontWeight.BOLD);
1531
1532                             // LOUITODO This might not work
1533                             conversationContentsPanel.add(postedBy);
1534                             //conversationContentsPanel.insert(postedBy, conversationPanel.getWidgetIndex
1535                             (replyToConversation));
1536
1537                             Label messageContents = new Label(messages[i][2]);
1538                             conversationContentsPanel.add(messageContents);
1539
1540                             conversationLastTimeStamp = Long.parseLong(msgs[i][1]);
1541                         }
1542                     }
1543
1544                     if(!refresh) {
1545                         conversationPanel.add(replyToConversation);
1546                         final FlowPanel conversationReplyPanel = new FlowPanel();
1547                         convoPanelSetup_input.setCharacterWidth(80);
1548                         convoPanelSetup_input.setVisibleLines(10);
1549                         conversationReplyPanel.add(convoPanelSetup_input);
1550
1551                         HorizontalPanel conversationReplyControlsPanel = new HorizontalPanel();
1552                         conversationReplyPanel.add(conversationReplyControlsPanel);
1553
1554                         Label stop = new Label("Page auto update paused");
1555                         stop.getElement().getStyle().setProperty("color", "#FF0000");
1556                         stop.getElement().getStyle().setProperty("paddingRight", "55px");
1557                         conversationReplyControlsPanel.add(stop);
1558                         stop.addClickHandler(new ClickHandler() {
1559                             public void onClick(ClickEvent event) {
1560                                 conversation(conversationID, false);
1561                             }
1562                         });
1563
1564                         Button cancel = new Button("Cancel");
1565                         conversationReplyControlsPanel.add(cancel);
1566
1567                         cancel.addClickHandler(new ClickHandler() {
1568                             public void onClick(ClickEvent event) {

```

```

1567         conversation(conversationID, false);
1568     }
1569     });
1570
1571     Button send = new Button("Send");
1572     conversationReplyControlsPanel.add(send);
1573     send.addClickHandler(new ClickHandler() {
1574         public void onClick(ClickEvent event) {
1575             turtlenet.sendMessageToCHAT(convoPanelSetup_input.getText(), convoPanelSetup_convosig, new
1576 AsyncCallback<String>() {
1577
1578                 public void onFailure(Throwable caught) {
1579                     System.out.println("turtlenet.sendMessageToCHAT failed: " + caught);
1580                 }
1581                 public void onSuccess(String postingSuccess) {
1582                     //Reload the conversation after the new message has been added
1583                     conversation(convoPanelSetup_convosig, false);
1584                 }
1585             });
1586         }
1587     });
1588
1589     replyToConversation.addClickHandler(new ClickHandler() {
1590         public void onClick(ClickEvent event) {
1591             location = "replyToConversation";
1592             refreshID = "";
1593         }
1594     });
1595     conversationPanel.add(conversationReplyPanel);
1596
1597     });
1598     });
1599     });
1600
1601     // Add style name for CSS
1602     conversationPanel.addStyleName("gwt-conversation");
1603 }
1604
1605 TextBox newGroup_nameInput = new TextBox();
1606 private void newGroup() {
1607     location = "newGroup";
1608     refreshID = "";
1609
1610     RootPanel.get().clear();
1611     navigation();
1612     FlexTable newGroupPanel = new FlexTable();
1613     RootPanel.get().add(newGroupPanel);
1614
1615     newGroupPanel.setWidget(0, 0, new Label("Category name: "));
1616     newGroupPanel.setWidget(0, 1, newGroup_nameInput);
1617
1618     Button createGroup = new Button("Create category");
1619     newGroupPanel.setWidget(0, 2, createGroup);
1620
1621     createGroup.addClickHandler(new ClickHandler() {
1622         public void onClick(ClickEvent event) {
1623             turtlenet.addCategory(newGroup_nameInput.getText(), new AsyncCallback<String>() {
1624                 public void onFailure(Throwable caught) {
1625                     System.out.println("turtlenet.addCategory failed: " + caught);
1626                 }
1627                 public void onSuccess(String result) {
1628                     //if (result.equals("success")) {
1629                     editGroup(newGroup_nameInput.getText());
1630                     //} else {
1631                     //System.out.println("turtlenet.addCategory onSuccess String result did not equal success");
1632                     //}
1633                 }
1634             });
1635         }
1636     });
1637
1638     newGroupPanel.addStyleName("gwt-new-group");
1639 }
1640
1641 private void editGroup(final String groupID) {
1642     location = "editGroup";
1643     refreshID = "";
1644
1645     FlexTable editGroupPanel = new FlexTable();
1646     editGroupPanel.clear();
1647     RootPanel.get().add(editGroupPanel);
1648
1649     editGroupPanel.setWidget(1, 0, new Label("Currently in category: "));
1650     final ListBox currentMembers = new ListBox();
1651     currentMembers.setVisibleItemCount(10);
1652     currentMembers.setWidth("150px");
1653     editGroupPanel.setWidget(1, 1, currentMembers);
1654
1655     turtlenet.getCategoryMembers(groupID, new AsyncCallback<String[][]>() {
1656         String[][] result;
1657         int i;
1658         public void onFailure(Throwable caught) {

```

```

1659         System.out.println("turtlenet.getCategoryMembers failed: " + caught);
1660     }
1661     public void onSuccess(String[][] _result) {
1662         result = _result;
1663         for (i = 0; i < result.length; i++) {
1664             currentMembers.addItem(result[i][0]);
1665             currentMembers.setValue(i, result[i][1]); //their key
1666         }
1667     }
1668 });
1669
1670 Button removeFromGroup = new Button("Remove from group");
1671 editGroupPanel.setWidget(1, 2, removeFromGroup);
1672 removeFromGroup.addClickHandler(new ClickHandler() {
1673     public void onClick(ClickEvent event) {
1674         AsyncCallback<String>() {
1675             public void onFailure(Throwable caught) {
1676                 System.out.println("turtlenet.removeFromCategory failed: " + caught);
1677             }
1678             public void onSuccess(String result) {
1679                 friendsList(groupID);
1680             }
1681         });
1682     }
1683 });
1684
1685 editGroupPanel.setWidget(2, 0, new Label("Add a friend: "));
1686 final List<String> allFriends = new ArrayList<>();
1687 allFriends.setVisibleItemCount(1);
1688 allFriends.setWidth("150px");
1689 editGroupPanel.setWidget(2, 1, allFriends);
1690
1691 turtlenet.getPeople(new AsyncCallback<String[][]>() {
1692     String[][] result;
1693     int i;
1694     public void onFailure(Throwable caught) {
1695         System.out.println("turtlenet.getPeople failed: " + caught);
1696     }
1697     public void onSuccess(String[][] _result) {
1698         result = _result;
1699         for (i = 0; i < result.length; i++) {
1700             String friendKey = new String(result[i][1]);
1701             allFriends.addItem(result[i][0]);
1702             allFriends.setValue(i, friendKey);
1703         }
1704     }
1705 });
1706
1707 Button addFriend = new Button("Add friend");
1708 editGroupPanel.setWidget(2, 2, addFriend);
1709 addFriend.addClickHandler(new ClickHandler() {
1710     public void onClick(ClickEvent event) {
1711         AsyncCallback<String>() {
1712             public void onFailure(Throwable caught) {
1713                 System.out.println("turtlenet.addToCategory failed: " + caught);
1714             }
1715             public void onSuccess(String result) {
1716                 //if (result.equals("success")) {
1717                     friendsList(groupID);
1718                 //} else {
1719                     //System.out.println("turtlenet.addToCategory onSuccess String result did not equal success");
1720                 //}
1721             }
1722         });
1723     }
1724 });
1725
1726 editGroupPanel.addStyleName("gwt-edit-group");
1727 }
1728
1729 TextBox addFriend_keyInput = new TextBox();
1730 private void addFriend() {
1731     location = "addFriend";
1732     refreshID = "";
1733
1734     RootPanel.get().clear();
1735     navigation();
1736     FlexTable addFriendPanel = new FlexTable();
1737     RootPanel.get().add(addFriendPanel);
1738
1739     addFriendPanel.setWidget(0, 0, new Label("Enter the key of the person you wish to add:"));
1740     addFriend_keyInput.setVisibleLength(100);
1741     addFriendPanel.setWidget(1, 0, addFriend_keyInput);
1742
1743     Button submit = new Button("Add key");
1744     submit.setWidth("640px");
1745     addFriendPanel.setWidget(2, 0, submit);
1746     final Label success = new Label("");
1747     addFriendPanel.setWidget(3, 0, success);
1748
1749     submit.addClickHandler(new ClickHandler() {

```

```
1750         public void onClick(ClickEvent event) {
1751             turtlenet.addKey(addFriend_keyInput.getText(), new AsyncCallback<String>() {
1752                 public void onFailure(Throwable caught) {
1753                     success.setText("Key could not be added");
1754                     System.out.println("turtlenet.addKey failed: " + caught);
1755                 }
1756                 public void onSuccess(String result) {
1757                     if (result.equals("success")) {
1758                         success.setText("Key has been added");
1759                     } else {
1760                         success.setText("Key could not be added");
1761                         System.out.println("turtlenet.addKey onSuccess String result did not equal success");
1762                     }
1763                 }
1764             });
1765         }
1766     });
1767     addFriendPanel.addStyleName("gwt-friend");
1768 }
1769 }
```