```java
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.transformation.FilteredList;
import javafx.collections.transformation.SortedList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.Label;
import javafx.scene.control.MenuItem;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class FileTableController implements Initializable {
    // From ConnectController
    @FXML
    private String serverHN, serverPort, userName, userHN, userPort, userSpeed;

    // MyFiles Table (Local files)
    @FXML
    private TableView<FileObject> myFilesTableView;
    @FXML
    private TableColumn<FileObject, String> myFilenameColumn;
    @FXML
    private TableColumn<FileObject, String> myDescColumn;

    // Server Files Table
    @FXML
    private TableView<FileObject> serverFilesTableView;
    @FXML
    private TableColumn<FileObject, String> servFilenameColumn;
    @FXML
    private TableColumn<FileObject, String> servHNColumn;
    @FXML
    private TableColumn<FileObject, String> servSpeedColumn;

    // General display attributes
    @FXML
    private Label close;
    @FXML
    private TextField searchField;
    @FXML
    private Label connectToLabel;
    @FXML
    private Label userLabel;
    @FXML
    private Label ipLabel;
    @FXML
    private Button disconnectBtn;

    // Description Editor attributes
    @FXML
    private AnchorPane editDescPane;
    @FXML
    private Label editDescLabel;
    @FXML
    private Button updateDescBtn;
```

```
70      @FXML
71      private TextArea editDescTextArea;
72
73      private User user;
74
75      // Data received from ConnectController
76      public void initData(User user, String serverHN, String serverPort, String
        userName, String userHN, String userPort,
77              String userSpeed) {
78          this.serverHN = serverHN;
79          this.serverPort = serverPort;
80          this.userName = userName;
81          this.userHN = userHN;
82          this.userPort = userPort;
83          this.userSpeed = userSpeed;
84
85          this.user = user;
86          // Set connection session attributes banner
87          sessionAttributes();
88      }
89
90      @Override
91      public void initialize(URL location, ResourceBundle resources) {
92          // Set up MyFile Table columns
93          myFilenameColumn.setCellValueFactory(new PropertyValueFactory<FileObject,
            String>("filename"));
94          myDescColumn.setCellValueFactory(new PropertyValueFactory<FileObject,
            String>("description"));
95
96          // Set up Server Files Table columns
97          servFilenameColumn.setCellValueFactory(new PropertyValueFactory<FileObject,
            String>("filename"));
98          servHNColumn.setCellValueFactory(new PropertyValueFactory<FileObject,
            String>("hostname"));
99          servSpeedColumn.setCellValueFactory(new PropertyValueFactory<FileObject,
            String>("speed"));
100
101         // Fill MyFiles Table with files in designated MyFiles folder
102         myFilesTableView.setItems(getMyFiles());
103         // Load dummy file data for Server Table
104         serverFilesTableView.setItems(getServerFiles());
105
106         // Search Server File Table for keywords
107         ObservableList<FileObject> serverFiles = getServerFiles();
108         FilteredList<FileObject> filteredData = new FilteredList(serverFiles);
109         SortedList<FileObject> fileSortedList = new SortedList<>(filteredData);
110         serverFilesTableView.setItems(fileSortedList);
111
            fileSortedList.comparatorProperty().bind(serverFilesTableView.comparatorProperty(
            ));
112         // Dynamic Server File Table search results based on Description, Filename,
113         // Hostname, and connection Speed.
114         searchField.textProperty()
115                 .addListener((observable, oldValue, newValue) ->
                    filteredData.setPredicate(
116                         f ->
                            f.getDescription().toLowerCase().contains(searchField.getText().t
                            oLowerCase())
117                                 ||
                                    f.getFilename().toLowerCase().contains(searchField.getTex
                                    t().toLowerCase())
118                                 || f.getHostname().contains(searchField.getText())
119                                 ||
                                    f.getSpeed().toLowerCase().contains(searchField.getText()
                                    .toLowerCase())));
120
121         // Right-click Menu on MyFiles Table (Description Editor)
122         MenuItem mi1 = new MenuItem("Edit Description");
123         mi1.setOnAction((ActionEvent event) -> {
```

```
            FileObject item = myFilesTableView.getSelectionModel().getSelectedItem();
            System.out.println("Selected item: " + item.getFilename());
            editDescription(item);
        });
        ContextMenu menu = new ContextMenu();
        menu.getItems().add(mi1);
        myFilesTableView.setContextMenu(menu);

        MenuItem mi12 = new MenuItem("Get File");
        mi12.setOnAction((ActionEvent event) -> {
            FileObject item = serverFilesTableView.getSelectionModel().getSelectedItem();
            System.out.println("Selected item: " + item.getFilename());
            transferFile(item);
        });
        ContextMenu menu2 = new ContextMenu();
        menu2.getItems().add(mi12);
        serverFilesTableView.setContextMenu(menu2);
    }

    // Returns ObservableList of all FileObject objects in MyFiles folder
    private ObservableList<FileObject> getMyFiles() {
        ObservableList<FileObject> files = FXCollections.observableArrayList();

        File folder = new File("./");
        File[] listOfFiles = folder.listFiles();

        assert listOfFiles != null;
        for (File file : listOfFiles) {
            if (file.isFile()) {
                files.add(new FileObject(file.getName(), "Add Description"));
            }
        }

        return files;
    }

    // Returns ObservableList of all FileObject objects used to fill Server File
    // Table
    // CURRENTLY -> Fills ObservableList with hardcoded dummy data
    // INTENDED -> Fill ObservableList with FileObjects gathered from "filelist.xml"
    // received from server
    private ObservableList<FileObject> getServerFiles() {

        ObservableList<FileObject> files = FXCollections.observableArrayList();

        return files;
    }

    // Handles Description edit of FileObject Description attribute
    private void editDescription(FileObject item) {
        // Set visible & fill with sought FileObject data
        editDescPane.setVisible(true);
        editDescLabel.setText("Edit Description of " + item.getFilename());
        editDescTextArea.setText("" + item.getDescription());

        // Handles "Update" button
        // Sets FileObject Description, sets Pane to invisible, refreshes (updates)
        // MyFile Table
        updateDescBtn.setOnAction((ActionEvent event) -> {
            System.out.println(editDescTextArea.getText());
            item.setDescription(editDescTextArea.getText());
            editDescPane.setVisible(false);
            myFilesTableView.refresh();
            System.out.println("Close Edit");
        });

    }

    private void transferFile(FileObject item) {
```

```
193             System.out.println("Transfer file: " + item.getFilename());
194         }
195 
196         private void sessionAttributes() {
197             connectToLabel.setText("Connected to " + serverHN + " on " + serverPort);
198             userLabel.setText("User: " + userName);
199             ipLabel.setText("IP: " + userHN);
200         }
201 
202         // Called when a user clicks the GetFile button. Grabs the text from the search
203         // bar and looks for filesnames that match.
204         // If a file is downloaded add it to the localfiles view.
205         public void getFile() {
206             try {
207                 System.out.println("Transfer file: Started");
208                 if (user.retrieve(searchField.getText())) {
209                     System.out.println("Transfer file: Finished");
210                     myFilesTableView.getItems().add(new FileObject(searchField.getText(),
                        "Add Description"));
211                     myFilesTableView.refresh();
212                 } else
213 
214                     // If the file wasnt downloaded an error occured.
215                     System.out.println("Transfer file: Error");
216             } catch (IOException e) {
217 
218                 e.printStackTrace();
219             }
220 
221         }
222 
223         public void disconnectBtnAction() {
224             System.out.println("Disconnect");
225         }
226 
227         // Called when a user clicks the searchServer button. Grabs the text from the
228         // search bar on the GUI to search file description for matches.
229         // The results from the search are uploaded to the serverFilesTableView.
230         public void searchServer() {
231             ObservableList<FileObject> files = FXCollections.observableArrayList();
232 
233             if (user.search(searchField.getText())) {
234                 ArrayList<AvailableFile> uFiles = user.getAvailableFiles();
235 
236                 for (AvailableFile file : uFiles) {
237                     System.out.println(file.fileName);
238                     files.add(new FileObject(file.fileName, file.hostName, file.speed, ""));
239                 }
240             }
241             FilteredList<FileObject> filteredFiles = new FilteredList(files);
242             SortedList<FileObject> fileSortedList = new SortedList<>(filteredFiles);
243             serverFilesTableView.setItems(fileSortedList);
244 
                fileSortedList.comparatorProperty().bind(serverFilesTableView.comparatorProperty(
                ));
245         }
246 
247         // Handles closing of window
248         public void closeBtnAction() {
249             user.quit();
250             Stage stage = (Stage) close.getScene().getWindow();
251             System.out.println("Application closed.");
252             stage.close();
253             System.exit(1);
254         }
255     }
256 
```