

UNIWERSYTET RZESZOWSKI
WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI



Bohdan Kudrenko

134935

Informatyka

System zarządzania uniwersytetem w technologii Java

Praca projektowa

Praca wykonana pod kierunkiem
mgr inż. Ewa Żesławska

Rzeszów 2025

Spis treści

1. Streszczenie w języku polskim i angielskim	3
2. Opis założeń projektu	4
3. Opis struktury projektu	5
3.1. Wykorzystane technologie	5
3.2. Struktura folderów, bazy danych i hierarchia klas.....	6
3.3. Minimalne wymagania sprzętowe i techniczne	8
4. Harmonogram realizacji projektu	9
5. Prezentacja warstwy użytkowej projektu.....	10
6. Podsumowanie	16
7. Oświadczenie studenta o samodzielności pracy	17
Spis rysunków	18
Spis listingów	19
Bibliografia	20

1. Streszczenie w języku polskim i angielskim

Projekt obejmuje stworzenie desktopowej aplikacji do zarządzania danymi studentów i wykładowców w systemie uniwersyteckim. Aplikacja została zbudowana w języku Java przy użyciu biblioteki Swing do stworzenia interfejsu graficznego oraz bazy danych SQLite do przechowywania danych. Użytkownik może przeglądać, dodawać, edytować i usuwać dane studentów i wykładowców. Dane są wyświetlane w tabelach z formularzami umożliwiającymi modyfikację, a wszystkie operacje są wykonywane przy użyciu wzorca DAO. Interfejs umożliwia wybór między modelem zarządzania studentami a modelem zarządzania wykładowcami. System automatycznie przekształca dane daty i obsługuje wyjątki SQL. Projekt zakłada przyszłe rozszerzenie o moduły kursów, ocen, planów zajęć oraz możliwość migracji do wersji webowej i wdrożenia systemu logowania z uprawnieniami.

The project involves the development of a desktop application for managing student and lecturer data within a university system. The application is built in Java using the Swing library for the graphical user interface and SQLite for data storage. Users can view, add, edit, and delete records of students and lecturers. Data is displayed in tables with editable forms, and all operations are implemented via the DAO design pattern. The interface allows switching between student and lecturer management modules. The system automatically handles date formatting and SQL exceptions. Future development plans include modules for course, grade, and schedule management, migration to a web version, and the implementation of user authentication with different access levels.

2. Opis założeń projektu

Celem projektu było stworzenie aplikacji desktopowej wspierającej zarządzanie podstawowymi danymi w systemie uniwersyteckim. Głównym problemem, który projekt stara się rozwiązać, jest brak centralizowanego i prostego w obsłudze narzędzia do administrowania informacjami o studentach i wykładowcach. Aplikacja ma na celu usprawnienie pracy administracyjnej poprzez automatyzację operacji CRUD (Create, Read, Update, Delete) na danych osobowych. Jest to istotne dla zapewnienia spójności i aktualności danych w uczelnianej bazie.

Wymagania funkcjonalne

- Umożliwia zarządzanie danymi studentów oraz wykładowców (dodawanie, edytowanie, usuwanie, przeglądanie).
- Umożliwia przeszukiwanie danych według imienia, nazwiska, e-maila i innych atrybutów.
- Obsługuje formularze do wprowadzania i aktualizacji danych osobowych.
- System automatycznie weryfikuje poprawność formatu dat (np. data ukończenia studiów).
- Wykładowcy i studenci są przechowywani w osobnych tabelach bazy danych.
- System pozwala na filtrowanie i sortowanie danych w tabelach.
- Zastosowano wzorzec DAO do komunikacji z bazą danych SQLite.
- Interfejs użytkownika umożliwia szybki dostęp do odpowiednich modułów (studenci / wykładowcy).

Wymagania niefunkcjonalne

- Interfejs użytkownika został zaprojektowany z wykorzystaniem biblioteki Swing i jest intuicyjny oraz prosty w obsłudze.
- System działa jako aplikacja desktopowa, nie wymagając połączenia z serwerem zewnętrznym.
- Obsługuje wyjątkowe sytuacje i błędy SQL w sposób przyjazny dla użytkownika.
- Aplikacja zapewnia spójność i aktualność danych dzięki bezpośredniemu dostępowi do lokalnej bazy danych.
- Architektura aplikacji umożliwia łatwą rozbudowę o nowe funkcje (np. zarządzanie kursami, planami zajęć).
- System został zoptymalizowany pod kątem wydajności przy pracy z lokalną bazą danych.

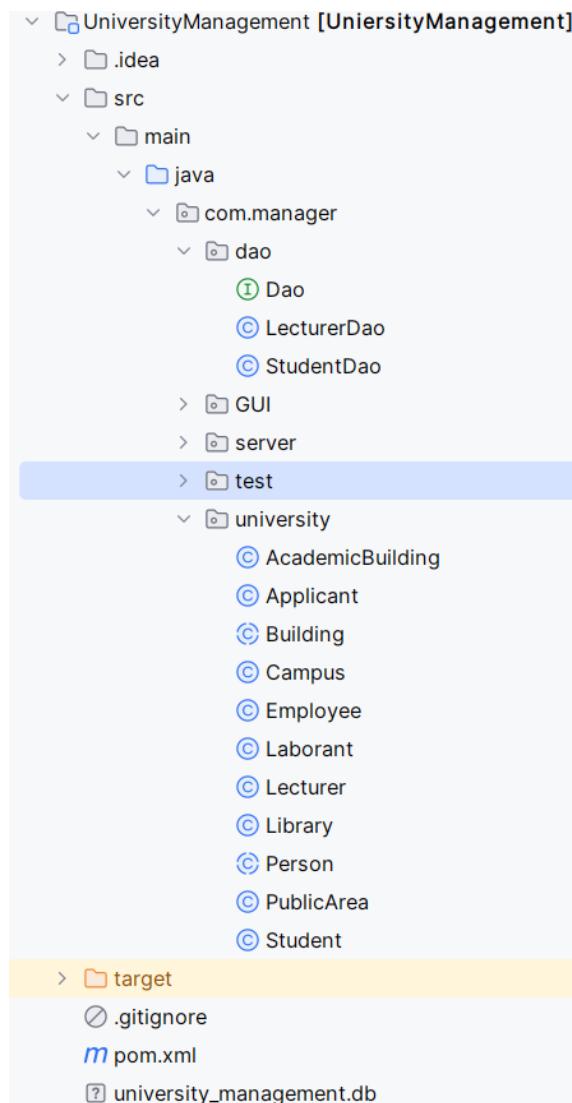
3. Opis struktury projektu

3.1. Wykorzystane technologie

Projekt został zrealizowany z wykorzystaniem następujących technologii:

- **Język programowania:** Java [4].
- **Interfejs graficzny użytkownika (GUI):** Java Swing.
- **Baza danych:** SQLite.
- **Sterownik bazy danych:** JDBC (Java Database Connectivity) do komunikacji między aplikacją a bazą danych [5].
- **Narzędzia:** LaTeX.

3.2. Struktura folderów, bazy danych i hierarchia klas



Rysunek 3.1. Struktura folderów projektu

Baza danych składa się z dwóch głównych tabel: students i lecturers. Poniżej przedstawiono schematy SQL użyte do ich utworzenia.

```

1 CREATE TABLE IF NOT EXISTS students (
2     id INTEGER PRIMARY KEY AUTOINCREMENT,
3     first_name TEXT NOT NULL,
4     last_name TEXT NOT NULL,
5     email TEXT,
6     major TEXT,
7     graduation_end DATE
8 );

```

Listing 3.1. Schemat SQL tabeli ‘students’

Źródło: Opracowanie własne na podstawie pliku DatabaseConnector.java.

```

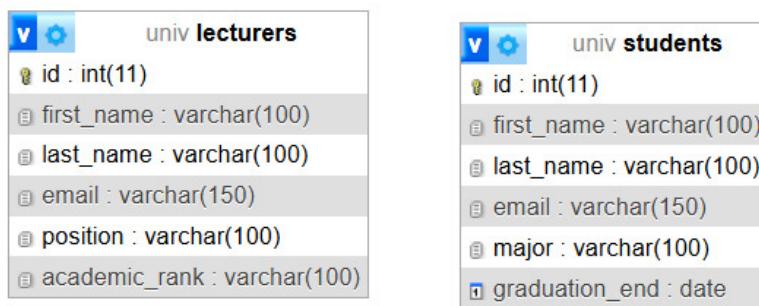
1 CREATE TABLE IF NOT EXISTS lecturers (
2     id INTEGER PRIMARY KEY AUTOINCREMENT,
3     first_name TEXT NOT NULL,
4     last_name TEXT NOT NULL,
5     email TEXT,
6     position TEXT,
7     academic_rank TEXT
8 );

```

Listing 3.2. Schemat SQL tabeli ‘lecturers’

Źródło: Opracowanie własne na podstawie pliku *DatabaseConnector.java*.

Na rysunku 3.2 przedstawiono diagram ERD (Entity-Relationship Diagram) dla bazy danych.

**Rysunek 3.2.** Diagram ERD bazy danych

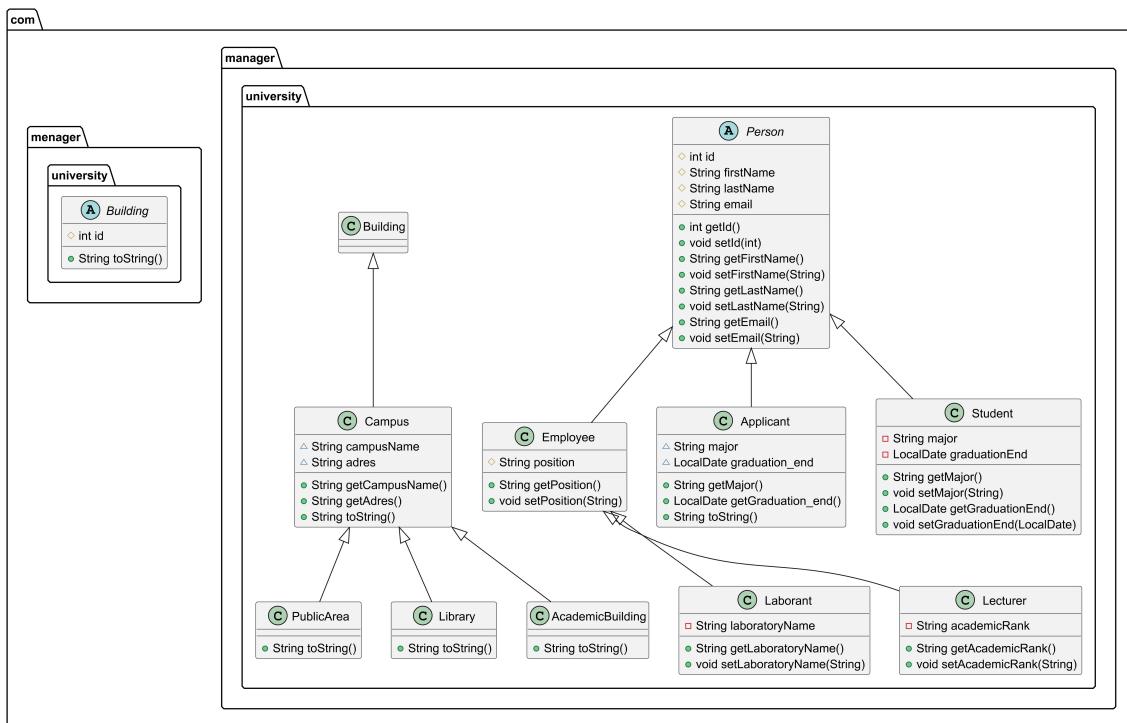
Źródło: Opracowanie własne.

Projekt opiera się na zasadach programowania obiektowego [1]. Stworzono hierarchię klas, która odzwierciedla strukturę danych w systemie.

- **Person:** Abstrakcyjna klasa bazowa, która zawiera wspólne atrybuty dla wszystkich osób w systemie, takie jak ID, imię, nazwisko i email.
- **Student:** Klasa dziedzicząca po ‘Person’, reprezentująca studenta. Dodatkowo zawiera pola ‘major’ (kierunek studiów) i ‘graduationEnd’ (data ukończenia studiów).
- **Applicant:** Klasa dziedzicząca po ‘Person’, reprezentująca kandydata. Dodatkowo zawiera pola ‘major’ (kierunek studiów) i ‘graduationEnd’ (data ukończenia studiów).
- **Employee:** Klasa dziedzicząca po ‘Person’, reprezentująca pracownika. Dodano pole ‘position’ (stanowisko).
- **Lecturer:** Klasa dziedzicząca po ‘Employee’, reprezentująca wykładowcę. Rozszerza klasę pracownika o pole ‘academicRank’ (stopień naukowy).

- **Laborant:** Klasa dziedzicząca po ‘Employee’, reprezentująca laboranta. Rozszerza klasę pracownika o pole ‘laboratoryName’ (nazwa laboratorium).
- **DAO (Data Access Object):** W projekcie zastosowano wzorzec projektowy DAO [3, 2]. Interfejs ‘Dao<T>’ definiuje podstawowe operacje bazodanowe, a jego implementacje ‘StudentDao’ i ‘LecturerDao’ zawierają logikę odpowiedzialną za komunikację z konkretnymi tabelami w bazie danych.

Na rysunku 3.3 przedstawiono również klasy *Building*, *Campus*, *PublicArea*, *Library*, *AcademicBuilding*.



Rysunek 3.3. Hierarchia klas
Źródło: Opracowanie własne.

3.3. Minimalne wymagania sprzętowe i techniczne

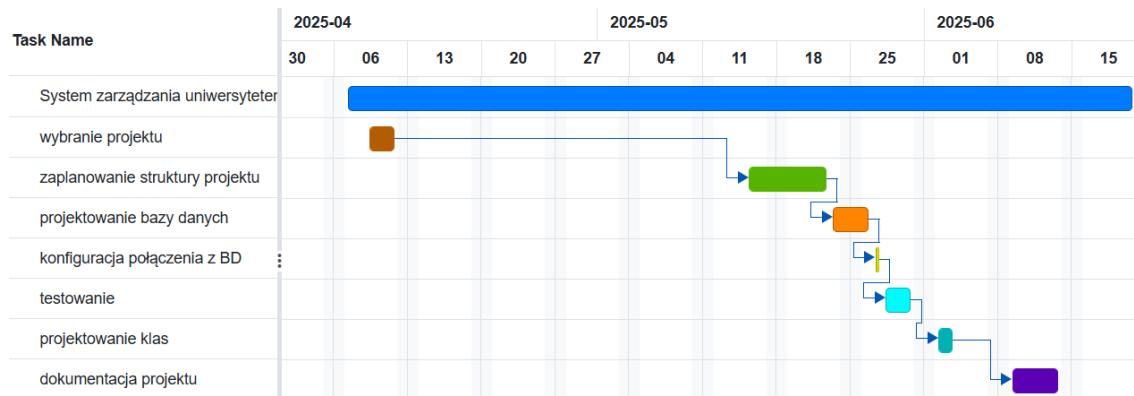
- **Procesor:** Intel Core i3 lub AMD Ryzen 3.
- **Pamięć RAM:** co najmniej 4 GB.
- **System operacyjny:** Windows / Linux.

Także warto pobrać IntelliJ IDEA Community Edition na stronie <https://www.jetbrains.com/idea/download/?section=windows>. Program wymaga JDK 21.

4. Harmonogram realizacji projektu

Link do repozytorium na GitHub <https://github.com/Bohdankudrenko/UniversityManagement>.

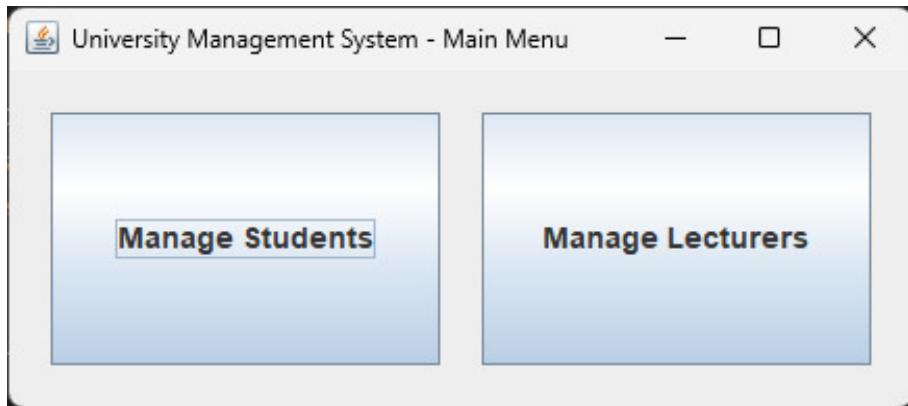
Realizacja projektu została zaplanowana i wykonana w określonych ramach czasowych. Podczas realizacji projektu napotkano pewne trudności, takie jak obsługa wyjątków SQL w kontekście interfejsu użytkownika oraz poprawne formatowanie daty między obiektami Javy a bazą danych. Problemy te zostały rozwiązane poprzez implementację globalnej obsługi wyjątków oraz zastosowanie klasy ‘LocalDate’.



Rysunek 4.1. Diagram Ganta

5. Prezentacja warstwy użytkowej projektu

Aplikacja posiada graficzny interfejs użytkownika (GUI) stworzony przy użyciu biblioteki Swing. Interfejs jest prosty i intuicyjny, co ułatwia zarządzanie danymi. Główne okno aplikacji (rys. 5.1) pozwala na wybór modułu do zarządzania — studentami lub wykładowcami.



Rysunek 5.1. Główne okno aplikacji
Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

Okno zarządzania studentami (rys. 5.2) zawiera tabelę z danymi, formularz do ich edycji i dodawania oraz przyciski do wykonywania operacji.

Analogicznie, okno zarządzania wykładowcami (rys. 5.3) oferuje te same funkcjonalności dla danych wykładowców.

```
1 package com.manager.GUI;
2
3 import com.manager.dao.StudentDao;
4 import com.manager.university.Student;
5
6 import javax.swing.*;
7 import javax.swing.table.DefaultTableModel;
8 import java.awt.*;
9 import java.sql.SQLException;
10 import java.time.LocalDate;
11 import java.time.format.DateTimeParseException;
12 import java.util.List;
13
14 public class StudentManagementApp extends JFrame {
15
16     private JTable studentTable;
17     private DefaultTableModel tableModel;
18     private JButton addButton, updateButton, deleteButton, clearButton;
19     private JTextField idField, firstNameField, lastNameField, emailField, majorField,
20     graduationEndField;
21     private StudentDao studentDao;
22
23     public StudentManagementApp() {
24         studentDao = new StudentDao();
25         setTitle("Student Management System");
26         setSize(800, 600);
27         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
28
29         // Create the student table
30         studentTable = new JTable(tableModel);
31         add(studentTable, "Center");
32
33         // Create buttons and fields
34         addButton = new JButton("Add");
35         updateButton = new JButton("Update");
36         deleteButton = new JButton("Delete");
37         clearButton = new JButton("Clear");
38
39         idField = new JTextField(10);
40         firstNameField = new JTextField(15);
41         lastNameField = new JTextField(15);
42         emailField = new JTextField(20);
43         majorField = new JTextField(10);
44         graduationEndField = new JTextField(10);
45
46         // Create a panel for buttons and fields
47         JPanel buttonPanel = new JPanel();
48         buttonPanel.add(addButton);
49         buttonPanel.add(updateButton);
50         buttonPanel.add(deleteButton);
51         buttonPanel.add(clearButton);
52
53         JPanel inputPanel = new JPanel();
54         inputPanel.add(idField);
55         inputPanel.add(firstNameField);
56         inputPanel.add(lastNameField);
57         inputPanel.add(emailField);
58         inputPanel.add(majorField);
59         inputPanel.add(graduationEndField);
60
61         // Add components to the frame
62         add(buttonPanel, "North");
63         add(inputPanel, "South");
64
65         // Set up the table model
66         tableModel = new DefaultTableModel();
67         tableModel.addColumn("ID");
68         tableModel.addColumn("First Name");
69         tableModel.addColumn("Last Name");
70         tableModel.addColumn("Email");
71         tableModel.addColumn("Major");
72         tableModel.addColumn("Graduation Date");
73
74         // Fill the table with initial data
75         List<Student> students = studentDao.getAllStudents();
76         for (Student student : students) {
77             Object[] row = {student.getId(), student.getFirstName(),
78                             student.getLastName(), student.getEmail(),
79                             student.getMajor(), student.getGraduationDate()};
80             tableModel.addRow(row);
81         }
82
83         // Set the table as the default component
84         setContentPane(studentTable);
85
86         // Set up event listeners
87         addButton.addActionListener(this);
88         updateButton.addActionListener(this);
89         deleteButton.addActionListener(this);
90         clearButton.addActionListener(this);
91
92         idField.addActionListener(this);
93         firstNameField.addActionListener(this);
94         lastNameField.addActionListener(this);
95         emailField.addActionListener(this);
96         majorField.addActionListener(this);
97         graduationEndField.addActionListener(this);
98
99         // Set up validation
100         firstNameField.setValidator(new JTextFieldValidator("First Name"));
101         lastNameField.setValidator(new JTextFieldValidator("Last Name"));
102         emailField.setValidator(new JTextFieldValidator("Email"));
103         majorField.setValidator(new JTextFieldValidator("Major"));
104         graduationEndField.setValidator(new JTextFieldValidator("Graduation Date"));
105
106         // Set up date pickers
107         graduationEndField.setDateFormatSymbols(new DateFormatSymbols());
108         graduationEndField.setFormatterFactory(new DateTextFormatter());
109
110         // Set up scroll bars
111         studentTable.setRowHeight(25);
112         studentTable.setFillsViewportHeight(true);
113
114         // Set up keyboard navigation
115         studentTable.setFocusable(true);
116         studentTable.requestFocus();
117
118         // Set up window events
119         addWindowListener(new WindowAdapter() {
120             public void windowClosing(WindowEvent e) {
121                 studentDao.close();
122             }
123         });
124
125         // Set up mouse events
126         studentTable.addMouseListener(new MouseAdapter() {
127             public void mouseClicked(MouseEvent e) {
128                 if (e.getClickCount() == 2) {
129                     int row = studentTable.getSelectedRow();
130                     if (row > -1) {
131                         updateButton.setEnabled(true);
132                         deleteButton.setEnabled(true);
133                     }
134                 }
135             }
136         });
137
138         // Set up key events
139         studentTable.addKeyListener(new KeyAdapter() {
140             public void keyPressed(KeyEvent e) {
141                 if (e.getKeyCode() == KeyEvent.VK_F1) {
142                     updateButton.setEnabled(true);
143                     deleteButton.setEnabled(true);
144                 }
145             }
146         });
147
148         // Set up context menu
149         studentTable.setContextMenuEnabled(true);
150
151         // Set up scroll bars
152         studentTable.setRowHeight(25);
153         studentTable.setFillsViewportHeight(true);
154
155         // Set up keyboard navigation
156         studentTable.setFocusable(true);
157         studentTable.requestFocus();
158
159         // Set up window events
160         addWindowListener(new WindowAdapter() {
161             public void windowClosing(WindowEvent e) {
162                 studentDao.close();
163             }
164         });
165
166         // Set up mouse events
167         studentTable.addMouseListener(new MouseAdapter() {
168             public void mouseClicked(MouseEvent e) {
169                 if (e.getClickCount() == 2) {
170                     int row = studentTable.getSelectedRow();
171                     if (row > -1) {
172                         updateButton.setEnabled(true);
173                         deleteButton.setEnabled(true);
174                     }
175                 }
176             }
177         });
178
179         // Set up key events
180         studentTable.addKeyListener(new KeyAdapter() {
181             public void keyPressed(KeyEvent e) {
182                 if (e.getKeyCode() == KeyEvent.VK_F1) {
183                     updateButton.setEnabled(true);
184                     deleteButton.setEnabled(true);
185                 }
186             }
187         });
188
189         // Set up context menu
190         studentTable.setContextMenuEnabled(true);
191
192         // Set up scroll bars
193         studentTable.setRowHeight(25);
194         studentTable.setFillsViewportHeight(true);
195
196         // Set up keyboard navigation
197         studentTable.setFocusable(true);
198         studentTable.requestFocus();
199
200         // Set up window events
201         addWindowListener(new WindowAdapter() {
202             public void windowClosing(WindowEvent e) {
203                 studentDao.close();
204             }
205         });
206
207         // Set up mouse events
208         studentTable.addMouseListener(new MouseAdapter() {
209             public void mouseClicked(MouseEvent e) {
210                 if (e.getClickCount() == 2) {
211                     int row = studentTable.getSelectedRow();
212                     if (row > -1) {
213                         updateButton.setEnabled(true);
214                         deleteButton.setEnabled(true);
215                     }
216                 }
217             }
218         });
219
220         // Set up key events
221         studentTable.addKeyListener(new KeyAdapter() {
222             public void keyPressed(KeyEvent e) {
223                 if (e.getKeyCode() == KeyEvent.VK_F1) {
224                     updateButton.setEnabled(true);
225                     deleteButton.setEnabled(true);
226                 }
227             }
228         });
229
230         // Set up context menu
231         studentTable.setContextMenuEnabled(true);
232
233         // Set up scroll bars
234         studentTable.setRowHeight(25);
235         studentTable.setFillsViewportHeight(true);
236
237         // Set up keyboard navigation
238         studentTable.setFocusable(true);
239         studentTable.requestFocus();
240
241         // Set up window events
242         addWindowListener(new WindowAdapter() {
243             public void windowClosing(WindowEvent e) {
244                 studentDao.close();
245             }
246         });
247
248         // Set up mouse events
249         studentTable.addMouseListener(new MouseAdapter() {
250             public void mouseClicked(MouseEvent e) {
251                 if (e.getClickCount() == 2) {
252                     int row = studentTable.getSelectedRow();
253                     if (row > -1) {
254                         updateButton.setEnabled(true);
255                         deleteButton.setEnabled(true);
256                     }
257                 }
258             }
259         });
260
261         // Set up key events
262         studentTable.addKeyListener(new KeyAdapter() {
263             public void keyPressed(KeyEvent e) {
264                 if (e.getKeyCode() == KeyEvent.VK_F1) {
265                     updateButton.setEnabled(true);
266                     deleteButton.setEnabled(true);
267                 }
268             }
269         });
270
271         // Set up context menu
272         studentTable.setContextMenuEnabled(true);
273
274         // Set up scroll bars
275         studentTable.setRowHeight(25);
276         studentTable.setFillsViewportHeight(true);
277
278         // Set up keyboard navigation
279         studentTable.setFocusable(true);
280         studentTable.requestFocus();
281
282         // Set up window events
283         addWindowListener(new WindowAdapter() {
284             public void windowClosing(WindowEvent e) {
285                 studentDao.close();
286             }
287         });
288
289         // Set up mouse events
290         studentTable.addMouseListener(new MouseAdapter() {
291             public void mouseClicked(MouseEvent e) {
292                 if (e.getClickCount() == 2) {
293                     int row = studentTable.getSelectedRow();
294                     if (row > -1) {
295                         updateButton.setEnabled(true);
296                         deleteButton.setEnabled(true);
297                     }
298                 }
299             }
300         });
301
302         // Set up key events
303         studentTable.addKeyListener(new KeyAdapter() {
304             public void keyPressed(KeyEvent e) {
305                 if (e.getKeyCode() == KeyEvent.VK_F1) {
306                     updateButton.setEnabled(true);
307                     deleteButton.setEnabled(true);
308                 }
309             }
310         });
311
312         // Set up context menu
313         studentTable.setContextMenuEnabled(true);
314
315         // Set up scroll bars
316         studentTable.setRowHeight(25);
317         studentTable.setFillsViewportHeight(true);
318
319         // Set up keyboard navigation
320         studentTable.setFocusable(true);
321         studentTable.requestFocus();
322
323         // Set up window events
324         addWindowListener(new WindowAdapter() {
325             public void windowClosing(WindowEvent e) {
326                 studentDao.close();
327             }
328         });
329
330         // Set up mouse events
331         studentTable.addMouseListener(new MouseAdapter() {
332             public void mouseClicked(MouseEvent e) {
333                 if (e.getClickCount() == 2) {
334                     int row = studentTable.getSelectedRow();
335                     if (row > -1) {
336                         updateButton.setEnabled(true);
337                         deleteButton.setEnabled(true);
338                     }
339                 }
340             }
341         });
342
343         // Set up key events
344         studentTable.addKeyListener(new KeyAdapter() {
345             public void keyPressed(KeyEvent e) {
346                 if (e.getKeyCode() == KeyEvent.VK_F1) {
347                     updateButton.setEnabled(true);
348                     deleteButton.setEnabled(true);
349                 }
350             }
351         });
352
353         // Set up context menu
354         studentTable.setContextMenuEnabled(true);
355
356         // Set up scroll bars
357         studentTable.setRowHeight(25);
358         studentTable.setFillsViewportHeight(true);
359
360         // Set up keyboard navigation
361         studentTable.setFocusable(true);
362         studentTable.requestFocus();
363
364         // Set up window events
365         addWindowListener(new WindowAdapter() {
366             public void windowClosing(WindowEvent e) {
367                 studentDao.close();
368             }
369         });
370
371         // Set up mouse events
372         studentTable.addMouseListener(new MouseAdapter() {
373             public void mouseClicked(MouseEvent e) {
374                 if (e.getClickCount() == 2) {
375                     int row = studentTable.getSelectedRow();
376                     if (row > -1) {
377                         updateButton.setEnabled(true);
378                         deleteButton.setEnabled(true);
379                     }
380                 }
381             }
382         });
383
384         // Set up key events
385         studentTable.addKeyListener(new KeyAdapter() {
386             public void keyPressed(KeyEvent e) {
387                 if (e.getKeyCode() == KeyEvent.VK_F1) {
388                     updateButton.setEnabled(true);
389                     deleteButton.setEnabled(true);
390                 }
391             }
392         });
393
394         // Set up context menu
395         studentTable.setContextMenuEnabled(true);
396
397         // Set up scroll bars
398         studentTable.setRowHeight(25);
399         studentTable.setFillsViewportHeight(true);
400
401         // Set up keyboard navigation
402         studentTable.setFocusable(true);
403         studentTable.requestFocus();
404
405         // Set up window events
406         addWindowListener(new WindowAdapter() {
407             public void windowClosing(WindowEvent e) {
408                 studentDao.close();
409             }
410         });
411
412         // Set up mouse events
413         studentTable.addMouseListener(new MouseAdapter() {
414             public void mouseClicked(MouseEvent e) {
415                 if (e.getClickCount() == 2) {
416                     int row = studentTable.getSelectedRow();
417                     if (row > -1) {
418                         updateButton.setEnabled(true);
419                         deleteButton.setEnabled(true);
420                     }
421                 }
422             }
423         });
424
425         // Set up key events
426         studentTable.addKeyListener(new KeyAdapter() {
427             public void keyPressed(KeyEvent e) {
428                 if (e.getKeyCode() == KeyEvent.VK_F1) {
429                     updateButton.setEnabled(true);
430                     deleteButton.setEnabled(true);
431                 }
432             }
433         });
434
435         // Set up context menu
436         studentTable.setContextMenuEnabled(true);
437
438         // Set up scroll bars
439         studentTable.setRowHeight(25);
440         studentTable.setFillsViewportHeight(true);
441
442         // Set up keyboard navigation
443         studentTable.setFocusable(true);
444         studentTable.requestFocus();
445
446         // Set up window events
447         addWindowListener(new WindowAdapter() {
448             public void windowClosing(WindowEvent e) {
449                 studentDao.close();
450             }
451         });
452
453         // Set up mouse events
454         studentTable.addMouseListener(new MouseAdapter() {
455             public void mouseClicked(MouseEvent e) {
456                 if (e.getClickCount() == 2) {
457                     int row = studentTable.getSelectedRow();
458                     if (row > -1) {
459                         updateButton.setEnabled(true);
460                         deleteButton.setEnabled(true);
461                     }
462                 }
463             }
464         });
465
466         // Set up key events
467         studentTable.addKeyListener(new KeyAdapter() {
468             public void keyPressed(KeyEvent e) {
469                 if (e.getKeyCode() == KeyEvent.VK_F1) {
470                     updateButton.setEnabled(true);
471                     deleteButton.setEnabled(true);
472                 }
473             }
474         });
475
476         // Set up context menu
477         studentTable.setContextMenuEnabled(true);
478
479         // Set up scroll bars
480         studentTable.setRowHeight(25);
481         studentTable.setFillsViewportHeight(true);
482
483         // Set up keyboard navigation
484         studentTable.setFocusable(true);
485         studentTable.requestFocus();
486
487         // Set up window events
488         addWindowListener(new WindowAdapter() {
489             public void windowClosing(WindowEvent e) {
490                 studentDao.close();
491             }
492         });
493
494         // Set up mouse events
495         studentTable.addMouseListener(new MouseAdapter() {
496             public void mouseClicked(MouseEvent e) {
497                 if (e.getClickCount() == 2) {
498                     int row = studentTable.getSelectedRow();
499                     if (row > -1) {
500                         updateButton.setEnabled(true);
501                         deleteButton.setEnabled(true);
502                     }
503                 }
504             }
505         });
506
507         // Set up key events
508         studentTable.addKeyListener(new KeyAdapter() {
509             public void keyPressed(KeyEvent e) {
510                 if (e.getKeyCode() == KeyEvent.VK_F1) {
511                     updateButton.setEnabled(true);
512                     deleteButton.setEnabled(true);
513                 }
514             }
515         });
516
517         // Set up context menu
518         studentTable.setContextMenuEnabled(true);
519
520         // Set up scroll bars
521         studentTable.setRowHeight(25);
522         studentTable.setFillsViewportHeight(true);
523
524         // Set up keyboard navigation
525         studentTable.setFocusable(true);
526         studentTable.requestFocus();
527
528         // Set up window events
529         addWindowListener(new WindowAdapter() {
530             public void windowClosing(WindowEvent e) {
531                 studentDao.close();
532             }
533         });
534
535         // Set up mouse events
536         studentTable.addMouseListener(new MouseAdapter() {
537             public void mouseClicked(MouseEvent e) {
538                 if (e.getClickCount() == 2) {
539                     int row = studentTable.getSelectedRow();
540                     if (row > -1) {
541                         updateButton.setEnabled(true);
542                         deleteButton.setEnabled(true);
543                     }
544                 }
545             }
546         });
547
548         // Set up key events
549         studentTable.addKeyListener(new KeyAdapter() {
550             public void keyPressed(KeyEvent e) {
551                 if (e.getKeyCode() == KeyEvent.VK_F1) {
552                     updateButton.setEnabled(true);
553                     deleteButton.setEnabled(true);
554                 }
555             }
556         });
557
558         // Set up context menu
559         studentTable.setContextMenuEnabled(true);
560
561         // Set up scroll bars
562         studentTable.setRowHeight(25);
563         studentTable.setFillsViewportHeight(true);
564
565         // Set up keyboard navigation
566         studentTable.setFocusable(true);
567         studentTable.requestFocus();
568
569         // Set up window events
570         addWindowListener(new WindowAdapter() {
571             public void windowClosing(WindowEvent e) {
572                 studentDao.close();
573             }
574         });
575
576         // Set up mouse events
577         studentTable.addMouseListener(new MouseAdapter() {
578             public void mouseClicked(MouseEvent e) {
579                 if (e.getClickCount() == 2) {
580                     int row = studentTable.getSelectedRow();
581                     if (row > -1) {
582                         updateButton.setEnabled(true);
583                         deleteButton.setEnabled(true);
584                     }
585                 }
586             }
587         });
588
589         // Set up key events
590         studentTable.addKeyListener(new KeyAdapter() {
591             public void keyPressed(KeyEvent e) {
592                 if (e.getKeyCode() == KeyEvent.VK_F1) {
593                     updateButton.setEnabled(true);
594                     deleteButton.setEnabled(true);
595                 }
596             }
597         });
598
599         // Set up context menu
600         studentTable.setContextMenuEnabled(true);
601
602         // Set up scroll bars
603         studentTable.setRowHeight(25);
604         studentTable.setFillsViewportHeight(true);
605
606         // Set up keyboard navigation
607         studentTable.setFocusable(true);
608         studentTable.requestFocus();
609
610         // Set up window events
611         addWindowListener(new WindowAdapter() {
612             public void windowClosing(WindowEvent e) {
613                 studentDao.close();
614             }
615         });
616
617         // Set up mouse events
618         studentTable.addMouseListener(new MouseAdapter() {
619             public void mouseClicked(MouseEvent e) {
620                 if (e.getClickCount() == 2) {
621                     int row = studentTable.getSelectedRow();
622                     if (row > -1) {
623                         updateButton.setEnabled(true);
624                         deleteButton.setEnabled(true);
625                     }
626                 }
627             }
628         });
629
630         // Set up key events
631         studentTable.addKeyListener(new KeyAdapter() {
632             public void keyPressed(KeyEvent e) {
633                 if (e.getKeyCode() == KeyEvent.VK_F1) {
634                     updateButton.setEnabled(true);
635                     deleteButton.setEnabled(true);
636                 }
637             }
638         });
639
640         // Set up context menu
641         studentTable.setContextMenuEnabled(true);
642
643         // Set up scroll bars
644         studentTable.setRowHeight(25);
645         studentTable.setFillsViewportHeight(true);
646
647         // Set up keyboard navigation
648         studentTable.setFocusable(true);
649         studentTable.requestFocus();
650
651         // Set up window events
652         addWindowListener(new WindowAdapter() {
653             public void windowClosing(WindowEvent e) {
654                 studentDao.close();
655             }
656         });
657
658         // Set up mouse events
659         studentTable.addMouseListener(new MouseAdapter() {
660             public void mouseClicked(MouseEvent e) {
661                 if (e.getClickCount() == 2) {
662                     int row = studentTable.getSelectedRow();
663                     if (row > -1) {
664                         updateButton.setEnabled(true);
665                         deleteButton.setEnabled(true);
666                     }
667                 }
668             }
669         });
670
671         // Set up key events
672         studentTable.addKeyListener(new KeyAdapter() {
673             public void keyPressed(KeyEvent e) {
674                 if (e.getKeyCode() == KeyEvent.VK_F1) {
675                     updateButton.setEnabled(true);
676                     deleteButton.setEnabled(true);
677                 }
678             }
679         });
680
681         // Set up context menu
682         studentTable.setContextMenuEnabled(true);
683
684         // Set up scroll bars
685         studentTable.setRowHeight(25);
686         studentTable.setFillsViewportHeight(true);
687
688         // Set up keyboard navigation
689         studentTable.setFocusable(true);
690         studentTable.requestFocus();
691
692         // Set up window events
693         addWindowListener(new WindowAdapter() {
694             public void windowClosing(WindowEvent e) {
695                 studentDao.close();
696             }
697         });
698
699         // Set up mouse events
700         studentTable.addMouseListener(new MouseAdapter() {
701             public void mouseClicked(MouseEvent e) {
702                 if (e.getClickCount() == 2) {
703                     int row = studentTable.getSelectedRow();
704                     if (row > -1) {
705                         updateButton.setEnabled(true);
706                         deleteButton.setEnabled(true);
707                     }
708                 }
709             }
710         });
711
712         // Set up key events
713         studentTable.addKeyListener(new KeyAdapter() {
714             public void keyPressed(KeyEvent e) {
715                 if (e.getKeyCode() == KeyEvent.VK_F1) {
716                     updateButton.setEnabled(true);
717                     deleteButton.setEnabled(true);
718                 }
719             }
720         });
721
722         // Set up context menu
723         studentTable.setContextMenuEnabled(true);
724
725         // Set up scroll bars
726         studentTable.setRowHeight(25);
727         studentTable.setFillsViewportHeight(true);
728
729         // Set up keyboard navigation
730         studentTable.setFocusable(true);
731         studentTable.requestFocus();
732
733         // Set up window events
734         addWindowListener(new WindowAdapter() {
735             public void windowClosing(WindowEvent e) {
736                 studentDao.close();
737             }
738         });
739
740         // Set up mouse events
741         studentTable.addMouseListener(new MouseAdapter() {
742             public void mouseClicked(MouseEvent e) {
743                 if (e.getClickCount() == 2) {
744                     int row = studentTable.getSelectedRow();
745                     if (row > -1) {
746                         updateButton.setEnabled(true);
747                         deleteButton.setEnabled(true);
748                     }
749                 }
750             }
751         });
752
753         // Set up key events
754         studentTable.addKeyListener(new KeyAdapter() {
755             public void keyPressed(KeyEvent e) {
756                 if (e.getKeyCode() == KeyEvent.VK_F1) {
757                     updateButton.setEnabled(true);
758                     deleteButton.setEnabled(true);
759                 }
760             }
761         });
762
763         // Set up context menu
764         studentTable.setContextMenuEnabled(true);
765
766         // Set up scroll bars
767         studentTable.setRowHeight(25);
768         studentTable.setFillsViewportHeight(true);
769
770         // Set up keyboard navigation
771         studentTable.setFocusable(true);
772         studentTable.requestFocus();
773
774         // Set up window events
775         addWindowListener(new WindowAdapter() {
776             public void windowClosing(WindowEvent e) {
777                 studentDao.close();
778             }
779         });
780
781         // Set up mouse events
782         studentTable.addMouseListener(new MouseAdapter() {
783             public void mouseClicked(MouseEvent e) {
784                 if (e.getClickCount() == 2) {
785                     int row = studentTable.getSelectedRow();
786                     if (row > -1) {
787                         updateButton.setEnabled(true);
788                         deleteButton.setEnabled(true);
789                     }
790                 }
791             }
792         });
793
794         // Set up key events
795         studentTable.addKeyListener(new KeyAdapter() {
796             public void keyPressed(KeyEvent e) {
797                 if (e.getKeyCode() == KeyEvent.VK_F1) {
798                     updateButton.setEnabled(true);
799                     deleteButton.setEnabled(true);
800                 }
801             }
802         });
803
804         // Set up context menu
805         studentTable.setContextMenuEnabled(true);
806
807         // Set up scroll bars
808         studentTable.setRowHeight(25);
809         studentTable.setFillsViewportHeight(true);
810
811         // Set up keyboard navigation
812         studentTable.setFocusable(true);
813         studentTable.requestFocus();
814
815         // Set up window events
816         addWindowListener(new WindowAdapter() {
817             public void windowClosing(WindowEvent e) {
818                 studentDao.close();
819             }
820         });
821
822         // Set up mouse events
823         studentTable.addMouseListener(new MouseAdapter() {
824             public void mouseClicked(MouseEvent e) {
825                 if (e.getClickCount() == 2) {
826                     int row = studentTable.getSelectedRow();
827                     if (row > -1) {
828                         updateButton.setEnabled(true);
829                         deleteButton.setEnabled(true);
830                     }
831                 }
832             }
833         });
834
835         // Set up key events
836         studentTable.addKeyListener(new KeyAdapter() {
837             public void keyPressed(KeyEvent e) {
838                 if (e.getKeyCode() == KeyEvent.VK_F1) {
839                     updateButton.setEnabled(true);
840                     deleteButton.setEnabled(true);
841                 }
842             }
843         });
844
845         // Set up context menu
846         studentTable.setContextMenuEnabled(true);
847
848         // Set up scroll bars
849         studentTable.setRowHeight(25);
850         studentTable.setFillsViewportHeight(true);
851
852         // Set up keyboard navigation
853         studentTable.setFocusable(true);
854         studentTable.requestFocus();
855
856         // Set up window events
857         addWindowListener(new WindowAdapter() {
858             public void windowClosing(WindowEvent e) {
859                 studentDao.close();
860             }
861         });
862
863         // Set up mouse events
864         studentTable.addMouseListener(new MouseAdapter() {
865             public void mouseClicked(MouseEvent e) {
866                 if (e.getClickCount() == 2) {
867                     int row = studentTable.getSelectedRow();
868                     if (row > -1) {
869                         updateButton.setEnabled(true);
870                         deleteButton.setEnabled(true);
871                     }
872                 }
873             }
874         });
875
876         // Set up key events
877         studentTable.addKeyListener(new KeyAdapter() {
878             public void keyPressed(KeyEvent e) {
879                 if (e.getKeyCode() == KeyEvent.VK_F1) {
880                     updateButton.setEnabled(true);
881                     deleteButton.setEnabled(true);
882                 }
883             }
884         });
885
886         // Set up context menu
887         studentTable.setContextMenuEnabled(true);
888
889         // Set up scroll bars
890         studentTable.setRowHeight(25);
891         studentTable.setFillsViewportHeight(true);
892
893         // Set up keyboard navigation
894         studentTable.setFocusable(true);
895         studentTable.requestFocus();
896
897         // Set up window events
898         addWindowListener(new WindowAdapter() {
899             public void windowClosing(WindowEvent e) {
900                 studentDao.close();
901             }
902         });
903
904         // Set up mouse events
905         studentTable.addMouseListener(new MouseAdapter() {
906             public void mouseClicked(MouseEvent e) {
907                 if (e.getClickCount() == 2) {
908                     int row = studentTable.getSelectedRow();
909                     if (row > -1) {
910                         updateButton.setEnabled(true);
911                         deleteButton.setEnabled(true);
912                     }
913                 }
914             }
915         });
916
917         // Set up key events
918         studentTable.addKeyListener(new KeyAdapter() {
919             public void keyPressed(KeyEvent e) {
920                 if (e.getKeyCode() == KeyEvent.VK_F1) {
921                     updateButton.setEnabled(true);
922                     deleteButton.setEnabled(true);
923                 }
924             }
925         });
926
927         // Set up context menu
928         studentTable.setContextMenuEnabled(true);
929
930         // Set up scroll bars
931         studentTable.setRowHeight(25);
932         studentTable.setFillsViewportHeight(true);
933
934         // Set up keyboard navigation
935         studentTable.setFocusable(true);
936         studentTable.requestFocus();
937
938         // Set up window events
939         addWindowListener(new WindowAdapter() {
940             public void windowClosing(WindowEvent e) {
941                 studentDao.close();
942             }
943         });
944
945         // Set up mouse events
946         studentTable.addMouseListener(new MouseAdapter() {
947             public void mouseClicked(MouseEvent e) {
948                 if (e.getClickCount() == 2) {
949                     int row = studentTable.getSelectedRow();
950                     if (row > -1) {
951                         updateButton.setEnabled(true);
952                         deleteButton.setEnabled(true);
953                     }
954                 }
955             }
956         });
957
958         // Set up key events
959         studentTable.addKeyListener(new KeyAdapter() {
960             public void keyPressed(KeyEvent e) {
961                 if (e.getKeyCode() == KeyEvent.VK_F1) {
962                     updateButton.setEnabled(true);
963                     deleteButton.setEnabled(true);
964                 }
965             }
966         });
967
968         // Set up context menu
969         studentTable.setContextMenuEnabled(true);
970
971         // Set up scroll bars
972         studentTable.setRowHeight(25);
973         studentTable.setFillsViewportHeight(true);
974
975         // Set up keyboard navigation
976         studentTable.setFocusable(true);
977         studentTable.requestFocus();
978
979         // Set up window events
980         addWindowListener(new WindowAdapter() {
981             public void windowClosing(WindowEvent e) {
982                 studentDao.close();
983             }
984         });
985
986         // Set up mouse events
987         studentTable.addMouseListener(new MouseAdapter() {
988             public void mouseClicked(MouseEvent e) {
989                 if (e.getClickCount() == 2) {
990                     int row = studentTable.getSelectedRow();
991                     if (row > -1) {
992                         updateButton.setEnabled(true);
993                         deleteButton.setEnabled(true);
994                     }
995                 }
996             }
997         });
998
999         // Set up key events
1000         studentTable.addKeyListener(new KeyAdapter() {
1001             public void keyPressed(KeyEvent e) {
1002                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1003                     updateButton.setEnabled(true);
1004                     deleteButton.setEnabled(true);
1005                 }
1006             }
1007         });
1008
1009         // Set up context menu
1010         studentTable.setContextMenuEnabled(true);
1011
1012         // Set up scroll bars
1013         studentTable.setRowHeight(25);
1014         studentTable.setFillsViewportHeight(true);
1015
1016         // Set up keyboard navigation
1017         studentTable.setFocusable(true);
1018         studentTable.requestFocus();
1019
1020         // Set up window events
1021         addWindowListener(new WindowAdapter() {
1022             public void windowClosing(WindowEvent e) {
1023                 studentDao.close();
1024             }
1025         });
1026
1027         // Set up mouse events
1028         studentTable.addMouseListener(new MouseAdapter() {
1029             public void mouseClicked(MouseEvent e) {
1030                 if (e.getClickCount() == 2) {
1031                     int row = studentTable.getSelectedRow();
1032                     if (row > -1) {
1033                         updateButton.setEnabled(true);
1034                         deleteButton.setEnabled(true);
1035                     }
1036                 }
1037             }
1038         });
1039
1040         // Set up key events
1041         studentTable.addKeyListener(new KeyAdapter() {
1042             public void keyPressed(KeyEvent e) {
1043                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1044                     updateButton.setEnabled(true);
1045                     deleteButton.setEnabled(true);
1046                 }
1047             }
1048         });
1049
1050         // Set up context menu
1051         studentTable.setContextMenuEnabled(true);
1052
1053         // Set up scroll bars
1054         studentTable.setRowHeight(25);
1055         studentTable.setFillsViewportHeight(true);
1056
1057         // Set up keyboard navigation
1058         studentTable.setFocusable(true);
1059         studentTable.requestFocus();
1060
1061         // Set up window events
1062         addWindowListener(new WindowAdapter() {
1063             public void windowClosing(WindowEvent e) {
1064                 studentDao.close();
1065             }
1066         });
1067
1068         // Set up mouse events
1069         studentTable.addMouseListener(new MouseAdapter() {
1070             public void mouseClicked(MouseEvent e) {
1071                 if (e.getClickCount() == 2) {
1072                     int row = studentTable.getSelectedRow();
1073                     if (row > -1) {
1074                         updateButton.setEnabled(true);
1075                         deleteButton.setEnabled(true);
1076                     }
1077                 }
1078             }
1079         });
1080
1081         // Set up key events
1082         studentTable.addKeyListener(new KeyAdapter() {
1083             public void keyPressed(KeyEvent e) {
1084                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1085                     updateButton.setEnabled(true);
1086                     deleteButton.setEnabled(true);
1087                 }
1088             }
1089         });
1090
1091         // Set up context menu
1092         studentTable.setContextMenuEnabled(true);
1093
1094         // Set up scroll bars
1095         studentTable.setRowHeight(25);
1096         studentTable.setFillsViewportHeight(true);
1097
1098         // Set up keyboard navigation
1099         studentTable.setFocusable(true);
1100         studentTable.requestFocus();
1101
1102         // Set up window events
1103         addWindowListener(new WindowAdapter() {
1104             public void windowClosing(WindowEvent e) {
1105                 studentDao.close();
1106             }
1107         });
1108
1109         // Set up mouse events
1110         studentTable.addMouseListener(new MouseAdapter() {
1111             public void mouseClicked(MouseEvent e) {
1112                 if (e.getClickCount() == 2) {
1113                     int row = studentTable.getSelectedRow();
1114                     if (row > -1) {
1115                         updateButton.setEnabled(true);
1116                         deleteButton.setEnabled(true);
1117                     }
1118                 }
1119             }
1120         });
1121
1122         // Set up key events
1123         studentTable.addKeyListener(new KeyAdapter() {
1124             public void keyPressed(KeyEvent e) {
1125                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1126                     updateButton.setEnabled(true);
1127                     deleteButton.setEnabled(true);
1128                 }
1129             }
1130         });
1131
1132         // Set up context menu
1133         studentTable.setContextMenuEnabled(true);
1134
1135         // Set up scroll bars
1136         studentTable.setRowHeight(25);
1137         studentTable.setFillsViewportHeight(true);
1138
1139         // Set up keyboard navigation
1140         studentTable.setFocusable(true);
1141         studentTable.requestFocus();
1142
1143         // Set up window events
1144         addWindowListener(new WindowAdapter() {
1145             public void windowClosing(WindowEvent e) {
1146                 studentDao.close();
1147             }
1148         });
1149
1150         // Set up mouse events
1151         studentTable.addMouseListener(new MouseAdapter() {
1152             public void mouseClicked(MouseEvent e) {
1153                 if (e.getClickCount() == 2) {
1154                     int row = studentTable.getSelectedRow();
1155                     if (row > -1) {
1156                         updateButton.setEnabled(true);
1157                         deleteButton.setEnabled(true);
1158                     }
1159                 }
1160             }
1161         });
1162
1163         // Set up key events
1164         studentTable.addKeyListener(new KeyAdapter() {
1165             public void keyPressed(KeyEvent e) {
1166                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1167                     updateButton.setEnabled(true);
1168                     deleteButton.setEnabled(true);
1169                 }
1170             }
1171         });
1172
1173         // Set up context menu
1174         studentTable.setContextMenuEnabled(true);
1175
1176         // Set up scroll bars
1177         studentTable.setRowHeight(25);
1178         studentTable.setFillsViewportHeight(true);
1179
1180         // Set up keyboard navigation
1181         studentTable.setFocusable(true);
1182         studentTable.requestFocus();
1183
1184         // Set up window events
1185         addWindowListener(new WindowAdapter() {
1186             public void windowClosing(WindowEvent e) {
1187                 studentDao.close();
1188             }
1189         });
1190
1191         // Set up mouse events
1192         studentTable.addMouseListener(new MouseAdapter() {
1193             public void mouseClicked(MouseEvent e) {
1194                 if (e.getClickCount() == 2) {
1195                     int row = studentTable.getSelectedRow();
1196                     if (row > -1) {
1197                         updateButton.setEnabled(true);
1198                         deleteButton.setEnabled(true);
1199                     }
1200                 }
1201             }
1202         });
1203
1204         // Set up key events
1205         studentTable.addKeyListener(new KeyAdapter() {
1206             public void keyPressed(KeyEvent e) {
1207                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1208                     updateButton.setEnabled(true);
1209                     deleteButton.setEnabled(true);
1210                 }
1211             }
1212         });
1213
1214         // Set up context menu
1215         studentTable.setContextMenuEnabled(true);
1216
1217         // Set up scroll bars
1218         studentTable.setRowHeight(25);
1219         studentTable.setFillsViewportHeight(true);
1220
1221         // Set up keyboard navigation
1222         studentTable.setFocusable(true);
1223         studentTable.requestFocus();
1224
1225         // Set up window events
1226         addWindowListener(new WindowAdapter() {
1227             public void windowClosing(WindowEvent e) {
1228                 studentDao.close();
1229             }
1230         });
1231
1232         // Set up mouse events
1233         studentTable.addMouseListener(new MouseAdapter() {
1234             public void mouseClicked(MouseEvent e) {
1235                 if (e.getClickCount() == 2) {
1236                     int row = studentTable.getSelectedRow();
1237                     if (row > -1) {
1238                         updateButton.setEnabled(true);
1239                         deleteButton.setEnabled(true);
1240                     }
1241                 }
1242             }
1243         });
1244
1245         // Set up key events
1246         studentTable.addKeyListener(new KeyAdapter() {
1247             public void keyPressed(KeyEvent e) {
1248                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1249                     updateButton.setEnabled(true);
1250                     deleteButton.setEnabled(true);
1251                 }
1252             }
1253         });
1254
1255         // Set up context menu
1256         studentTable.setContextMenuEnabled(true);
1257
1258         // Set up scroll bars
1259         studentTable.setRowHeight(25);
1260         studentTable.setFillsViewportHeight(true);
1261
1262         // Set up keyboard navigation
1263         studentTable.setFocusable(true);
1264         studentTable.requestFocus();
1265
1266         // Set up window events
1267         addWindowListener(new WindowAdapter() {
1268             public void windowClosing(WindowEvent e) {
1269                 studentDao.close();
1270             }
1271         });
1272
1273         // Set up mouse events
1274         studentTable.addMouseListener(new MouseAdapter() {
1275             public void mouseClicked(MouseEvent e) {
1276                 if (e.getClickCount() == 2) {
1277                     int row = studentTable.getSelectedRow();
1278                     if (row > -1) {
1279                         updateButton.setEnabled(true);
1280                         deleteButton.setEnabled(true);
1281                     }
1282                 }
1283             }
1284         });
1285
1286         // Set up key events
1287         studentTable.addKeyListener(new KeyAdapter() {
1288             public void keyPressed(KeyEvent e) {
1289                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1290                     updateButton.setEnabled(true);
1291                     deleteButton.setEnabled(true);
1292                 }
1293             }
1294         });
1295
1296         // Set up context menu
1297         studentTable.setContextMenuEnabled(true);
1298
1299         // Set up scroll bars
1300         studentTable.setRowHeight(25);
1301         studentTable.setFillsViewportHeight(true);
1302
1303         // Set up keyboard navigation
1304         studentTable.setFocusable(true);
1305         studentTable.requestFocus();
1306
1307         // Set up window events
1308         addWindowListener(new WindowAdapter() {
1309             public void windowClosing(WindowEvent e) {
1310                 studentDao.close();
1311             }
1312         });
1313
1314         // Set up mouse events
1315         studentTable.addMouseListener(new MouseAdapter() {
1316             public void mouseClicked(MouseEvent e) {
1317                 if (e.getClickCount() == 2) {
1318                     int row = studentTable.getSelectedRow();
1319                     if (row > -1) {
1320                         updateButton.setEnabled(true);
1321                         deleteButton.setEnabled(true);
1322                     }
1323                 }
1324             }
1325         });
1326
1327         // Set up key events
1328         studentTable.addKeyListener(new KeyAdapter() {
1329             public void keyPressed(KeyEvent e) {
1330                 if (e.getKeyCode() == KeyEvent.VK_F1) {
1331                     updateButton.setEnabled(true);
1332                     deleteButton.setEnabled(true);
1333                 }
1334             }
1335         });
1336
1337         // Set up context menu
1338         studentTable.setContextMenuEnabled(true);
1339
1340         // Set up scroll bars
1341         studentTable.setRowHeight(25);
1342         studentTable.setFillsViewportHeight(true);
1343
1344         // Set up keyboard navigation
1345         studentTable.setFocusable(true);
1346         studentTable.requestFocus();
1347
1348         // Set up window events
1349         addWindowListener(new WindowAdapter() {
1350             public void windowClosing(WindowEvent e) {
1351                 studentDao.close();
1352             }
1353         });
1354
1355         // Set up mouse events
1356         studentTable.addMouseListener(new MouseAdapter() {
1357             public void mouseClicked(MouseEvent e) {
1358                 if (e.getClickCount() == 2) {
1359                     int row = studentTable.getSelectedRow();
1360                     if (row > -1) {
1361                         updateButton.setEnabled(true);
1362                         deleteButton.setEnabled(true);
1363                     }
1364                 }
1365             }
1366         });
1367
136
```

The screenshot shows a Java Swing application window titled "Student Management System". At the top, there is a JTable with columns: ID, First Name, Last Name, Email, Major, and Graduation Date. The table contains approximately 20 rows of student information. Below the table is a form panel with five text input fields labeled "ID:", "First Name:", "Last Name:", "Email:", and "Major:". At the bottom of the form panel are four buttons: "Add New", "Update Selected", "Delete Selected", and "Clear Form".

ID	First Name	Last Name	Email	Major	Graduation Date
31	Petro	Petrenko	student.petrenko.31...	Philology	2028-12-02
41	Oleksandr	Bondarenko	student.bondarenko...	Law	2028-11-13
5	Kateryna	Bondarenko	student.bondarenko...	Computer Science	2028-10-12
26	Maria	Romanenko	student.romanenko...	Economics	2028-08-05
40	Sofia	Romanenko	student.romanenko...	Law	2028-06-02
21	Sofia	Bondarenko	student.bondarenko...	Engineering	2028-02-14
12	Petro	Sydorenko	student.sydorenko.1...	Computer Science	2028-01-27
34	Petro	Ivanenko	studentivanenko.34...	Philology	2028-01-16
7	Sofia	Bondarenko	student.bondarenko...	Medicine	2027-11-25
48	Kateryna	Petrenko	student.petrenko.48...	Engineering	2027-10-24
3	Petro	Ivanenko	studentivanenko.3@...	Economics	2027-10-15
28	Oleksandr	Bondarenko	student.bondarenko...	Economics	2027-09-24
22	Maria	Ivanenko	studentivanenko.22...	Medicine	2027-09-17
50	Oleksandr	Romanenko	studentromanenko...	Economics	2027-08-26
29	Sofia	Bondarenko	student.bondarenko...	Medicine	2027-08-20
2	Petro	Kovalchuk	student.kovalchuk.2...	Engineering	2027-08-10
17	Petro	Bondarenko	student.bondarenko...	Computer Science	2027-07-08
44	Ivan	Bondarenko	student.bondarenko...	Economics	2027-05-28
37	Sofia	Ivanenko	studentivanenko.37...	Medicine	2027-04-18
16	Oleksandr	Petrenko	student.petrenko.16...	Law	2027-04-11
43	Oleksandr	Bondarenko	student.bondarenko...	Computer Science	2027-02-09
4	Maria	Ivanenko	student ivanenko 4@...	Medicine	2026-12-08

ID:
First Name:
Last Name:
Email:
Major:
Graduation Date (YYYY-MM-DD):

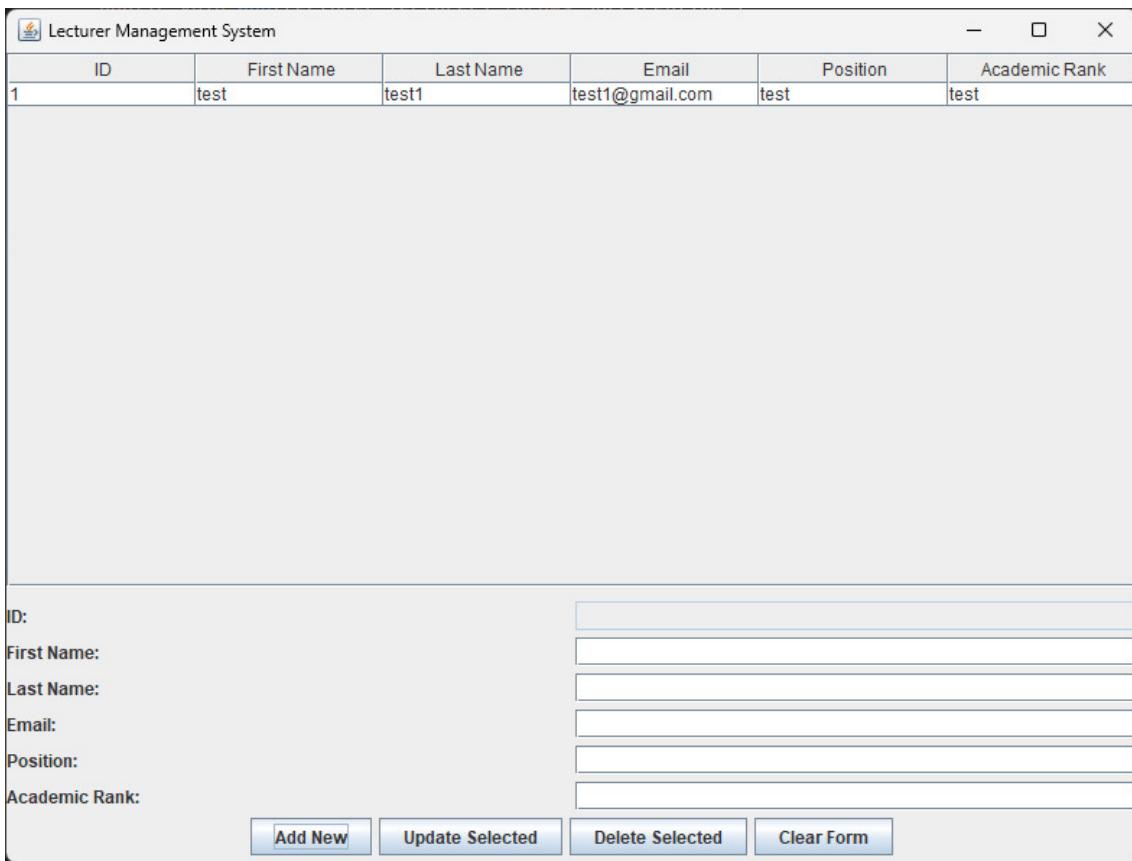
Add New Update Selected Delete Selected Clear Form

Rysunek 5.2. Interfejs do zarządzania studentami*Źródło: Zrzut ekranu z aplikacji (opracowanie własne).*

```

27     setLocationRelativeTo(null);
28     initUI();
29     refreshTableData();
30 }
31
32 private void initUI() {
33     tableModel = new DefaultTableModel(new Object[]{"ID", "First Name", "Last Name",
34         "Email", "Major", "Graduation Date"}, 0);
35     studentTable = new JTable(tableModel);
36     studentTable.setAutoCreateRowSorter(true);
37     studentTable.getSelectionModel().addListSelectionListener(e -> {
38         if (!e.getValueIsAdjusting()) {
39             fillFormFromSelectedRow();
40         }
41     });
42     JPanel formPanel = new JPanel(new GridLayout(6, 2, 5, 5));
43     idField = new JTextField();
44     idField.setEditable(false);
45     firstNameField = new JTextField();
46     lastNameField = new JTextField();
47     emailField = new JTextField();
48     majorField = new JTextField();
49     graduationEndField = new JTextField();
50     formPanel.add(new JLabel("ID:"));
51     formPanel.add(idField);
52     formPanel.add(new JLabel("First Name:"));

```

**Rysunek 5.3.** Interfejs do zarządzania wykładowcami

Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

```

53     formPanel.add(new JLabel("Last Name:"));
54     formPanel.add(lastNameField);
55     formPanel.add(new JLabel("Email:"));
56     formPanel.add(emailField);
57     formPanel.add(new JLabel("Major:"));
58     formPanel.add(majorField);
59     formPanel.add(new JLabel("Graduation Date (YYYY-MM-DD):"));
60     formPanel.add(graduationEndField);
61     JPanel buttonPanel = new JPanel();
62     addButton = new JButton("Add New");
63     updateButton = new JButton("Update Selected");
64     deleteButton = new JButton("Delete Selected");
65     clearButton = new JButton("Clear Form");
66     buttonPanel.add(addButton);
67     buttonPanel.add(updateButton);
68     buttonPanel.add(deleteButton);
69     buttonPanel.add(clearButton);
70     JPanel controlPanel = new JPanel(new BorderLayout());
71     controlPanel.add(formPanel, BorderLayout.CENTER);
72     controlPanel.add(buttonPanel, BorderLayout.SOUTH);
73     setLayout(new BorderLayout(10, 10));
74     add(new JScrollPane(studentTable), BorderLayout.CENTER);
75     add(controlPanel, BorderLayout.SOUTH);
76     addButton.addActionListener(e -> addStudent());
77     updateButton.addActionListener(e -> updateStudent());
78     deleteButton.addActionListener(e -> deleteStudent());
79     clearButton.addActionListener(e -> clearForm());

```

```
80     }
81
82     private void refreshTableData() {
83         tableView.setModel(tableModel);
84         try {
85             List<Student> students = studentDao.getAll();
86             for (Student student : students) {
87                 tableModel.addRow(new Object[] {
88                     student.getId(),
89                     student.getFirstName(),
90                     student.getLastName(),
91                     student.getEmail(),
92                     student.getMajor(),
93                     student.getGraduationEnd()
94                 });
95             }
96         } catch (SQLException e) {
97             JOptionPane.showMessageDialog(this, "Error loading data from DB: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
98         }
99     }
100
101     private void fillFormFromSelectedRow() {
102         int selectedRow = studentTable.getSelectedRow();
103         if (selectedRow != -1) {
104             idField.setText(tableModel.getValueAt(selectedRow, 0).toString());
105             firstNameField.setText(tableModel.getValueAt(selectedRow, 1).toString());
106             lastNameField.setText(tableModel.getValueAt(selectedRow, 2).toString());
107             emailField.setText(tableModel.getValueAt(selectedRow, 3).toString());
108             majorField.setText(tableModel.getValueAt(selectedRow, 4).toString());
109             graduationEndField.setText(tableModel.getValueAt(selectedRow, 5) != null ? tableModel.getValueAt(selectedRow, 5).toString() : "");
110         }
111     }
112
113     private void addStudent() {
114         try {
115             String firstName = firstNameField.getText();
116             String lastName = lastNameField.getText();
117             if (firstName.isEmpty() || lastName.isEmpty()) {
118                 JOptionPane.showMessageDialog(this, "First name and last name are required.", "Validation Error", JOptionPane.WARNING_MESSAGE);
119                 return;
120             }
121             Student student = new Student();
122             student.setFirstName(firstName);
123             student.setLastName(lastName);
124             student.setEmail(emailField.getText());
125             student.setMajor(majorField.getText());
126             student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));
127
128             studentDao.add(student);
129
130             refreshTableData();
131             clearForm();
132             JOptionPane.showMessageDialog(this, "Student added successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);
133         } catch (DateTimeParseException e) {
```

```
135     JOptionPane.showMessageDialog(this, "Invalid date format. Please use YYYY-MM
136 -DD.", "Format Error", JOptionPane.ERROR_MESSAGE);
137 } catch (SQLException e) {
138     JOptionPane.showMessageDialog(this, "Database error during add operation: "
139 + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
140 } catch (Exception e) {
141     JOptionPane.showMessageDialog(this, "An unknown error occurred: " + e.
142 getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
143 }
144
145 private void updateStudent() {
146     int selectedRow = studentTable.getSelectedRow();
147     if (selectedRow == -1) {
148         JOptionPane.showMessageDialog(this, "Please select a student to update.", "Error",
149             JOptionPane.WARNING_MESSAGE);
150         return;
151     }
152     try {
153         Student student = new Student();
154         student.setId(Integer.parseInt(idField.getText()));
155         student.setFirstName(firstNameField.getText());
156         student.setLastName(lastNameField.getText());
157         student.setEmail(emailField.getText());
158         student.setMajor(majorField.getText());
159         student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));
160
161         studentDao.update(student);
162         refreshTableData();
163         JOptionPane.showMessageDialog(this, "Student data updated successfully!", "Success",
164             JOptionPane.INFORMATION_MESSAGE);
165     } catch (SQLException e) {
166         JOptionPane.showMessageDialog(this, "Database error during update operation:
167 " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
168     } catch (Exception e) {
169         JOptionPane.showMessageDialog(this, "An error occurred during update: " + e.
170 getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
171     }
172 }
173
174 private void deleteStudent() {
175     int selectedRow = studentTable.getSelectedRow();
176     if (selectedRow == -1) { return; }
177     int confirmation = JOptionPane.showConfirmDialog(this, "Are you sure?", "Confirmation",
178         JOptionPane.YES_NO_OPTION);
179     if (confirmation == JOptionPane.YES_OPTION) {
180         try {
181             int id = Integer.parseInt(idField.getText());
182             studentDao.delete(id);
183             refreshTableData();
184             clearForm();
185             JOptionPane.showMessageDialog(this, "Student deleted successfully!", "Success",
186                 JOptionPane.INFORMATION_MESSAGE);
187         } catch (SQLException e) {
188             JOptionPane.showMessageDialog(this, "Database error during delete
189 operation: " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
190         }
191     }
192 }
```

```
184  
185     private void clearForm() {  
186         idField.setText("");  
187         firstNameField.setText("");  
188         lastNameField.setText("");  
189         emailField.setText("");  
190         majorField.setText("");  
191         graduationEndField.setText("");  
192         studentTable.clearSelection();  
193     }  
194  
195     public static void main(String[] args) {  
196         SwingUtilities.invokeLater(() -> {  
197             com.manager.server.DatabaseConnector.initializeDatabase();  
198             StudentManagementApp app = new StudentManagementApp();  
199             app.setVisible(true);  
200         });  
201     }  
202 }
```

Listing 5.1. Kod implementujący przycisk "Add New" w oknie zarządzania studentami.

6. Podsumowanie

W ramach niniejszej pracy zaprojektowano i zaimplementowano aplikację desktopową do zarządzania danymi studentów i wykładowców w systemie uniwersyteckim. Projekt został zrealizowany w technologii Java z wykorzystaniem biblioteki Swing do budowy interfejsu graficznego oraz bazy danych SQLite do przechowywania informacji. Główne cele pracy, takie jak stworzenie funkcjonalnego interfejsu CRUD (Create, Read, Update, Delete), implementacja warstwy dostępu do danych (DAO) oraz zaprojektowanie przezrzystej struktury klas, zostały w pełni osiągnięte. Aplikacja umożliwia dodawanie, edycję, usuwanie i przeglądanie danych w sposób intuicyjny dla użytkownika. Możliwe kierunki dalszego rozwoju projektu obejmują:

- Rozbudowę systemu o nowe moduły, np. zarządzanie kursami, ocenami czy planami zajęć.
- Implementację systemu uwierzytelniania użytkowników z różnymi poziomami uprawnień (np. administrator, pracownik dziekanatu).
- Migrację aplikacji do technologii webowej (np. z użyciem frameworka Spring Boot), co umożliwiający dostęp do systemu z dowolnego miejsca przez przeglądarkę internetową.
- Dodanie bardziej zaawansowanych funkcji, takich jak generowanie raportów i statystyk.

7. Oświadczenie studenta o samodzielności pracy

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OSWIADCZENIE STUDENTA O SAMODZIELNOSCI PRACY

.....Bohdan Kudrenko.....
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

.....Informatyka.....
Nazwa kierunku

.....134935.....
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: System zarządzania uniwersytetem w technologii Java
 - 1) została przygotowana przeze mnie samodzielnie*,
 - 2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
 - 3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
 - 4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/nie wyrażam zgody** na udostępnienie mojej pracy projektowej do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów 11.06.2025
(miejscowość, data)

Bohdan Kudrenko
(czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić

Spis rysunków

3.1	Struktura folderów projektu	6
3.2	Diagram ERD bazy danych	7
3.3	Hierarchia klas	8
4.1	Diagram Ganta	9
5.1	Główne okno aplikacji	10
5.2	Interfejs do zarządzania studentami	11
5.3	Interfejs do zarządzania wykładowcami	12

Listings

3.1	Schemat SQL tabeli ‘students‘	6
3.2	Schemat SQL tabeli ‘lecturers‘	7
5.1	Kod implementujący przycisk "Add New" w oknie zarządzania studentami.	10

Bibliografia

- [1] Joshua Bloch. *Effective Java*. Addison-Wesley Professional, 2018.
- [2] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.
- [3] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.
- [4] Cay S. Horstmann. *Core Java Volume I–Fundamentals*. Prentice Hall, 2019.
- [5] Abraham Silberschatz, Henry F. Korth, S. Sudarshan. *Database System Concepts*. McGraw-Hill, 2011.