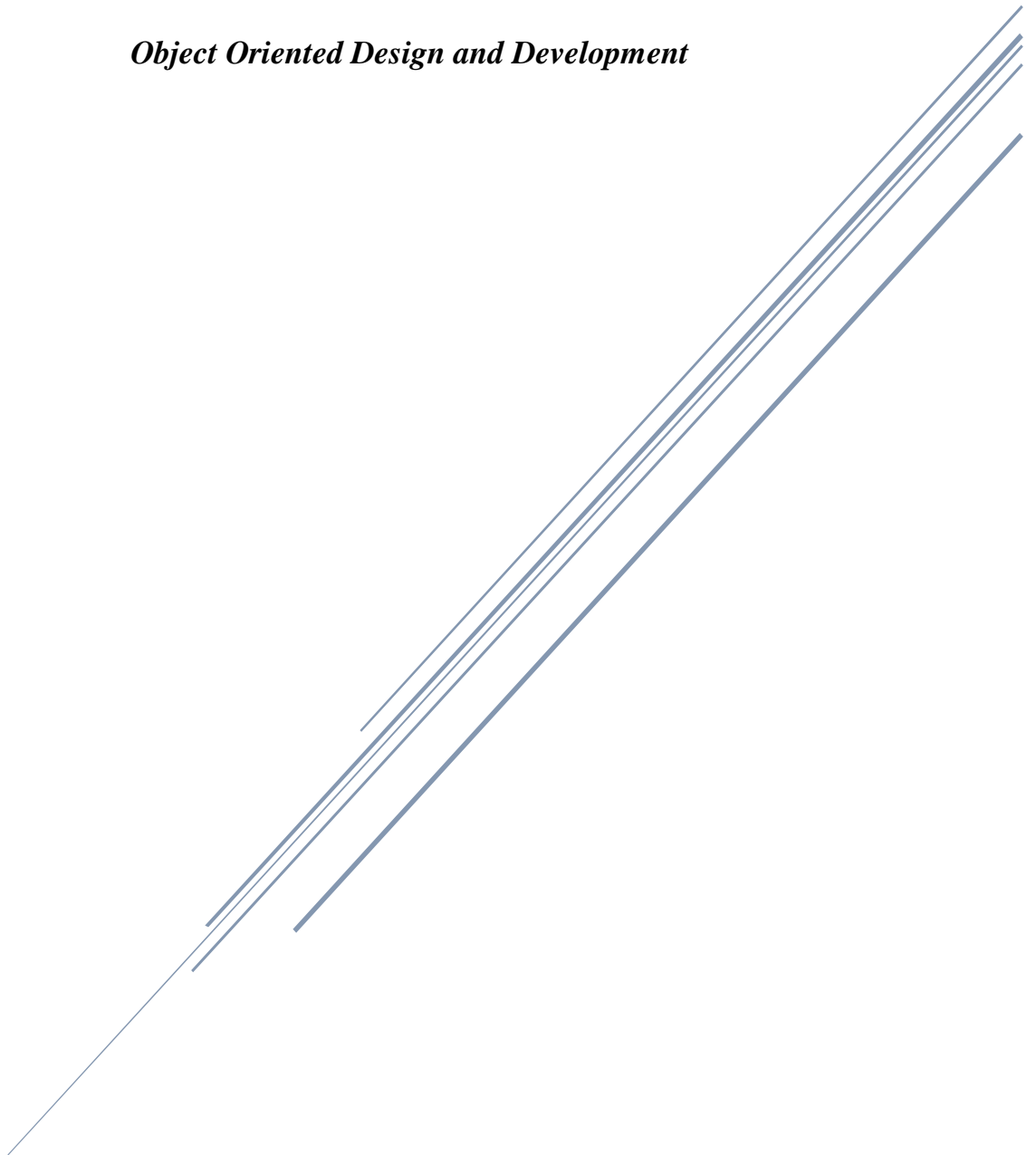


SOLENT

UNIVERSITY

Object Oriented Design and Development



Irina Condrat
10424186 / QHO543

Table of contents

| | |
|--|----|
| <i>Object Oriented Design and Development</i> | 0 |
| Table of Figures | 2 |
| 1. Introduction | 3 |
| 1.1. Project objective..... | 3 |
| 1.2. Used technologies | 4 |
| 2. Architecture of the System and Design of Classes: | 4 |
| 2.1. Class Structure Overview: | 5 |
| 2.2. Use Case Diagram..... | 6 |
| 2.3. The sequence diagram: | 7 |
| 2.4. The robustness diagram:..... | 8 |
| 3. Execution and Functionality Advancement | 9 |
| 3.1. Overview of Adapted Code from Group Project | 9 |
| 3.2. Student's perspective Functionality | 10 |
| 3.3. Admin's perspective functionality | 18 |
| 3.4. The new code developed for AE2 | 26 |
| 4. Principles of Software Engineering | 27 |
| 4.1. Design modularity and separation of concerns | 27 |
| 4.2. MVC Implementation | 29 |
| 4.3. Validation and error handling..... | 31 |
| 5. Evaluation and testing..... | 31 |
| 5.1. Test plan Junit..... | 32 |
| 5.2. Functionality testing with Tomcat | 32 |
| 5.3. Verification of Functions and Corrections | 34 |
| 6. Conclusions and possible future improvements | 36 |
| 6.1. Summary of achievements..... | 36 |
| 6.2. Possible improvements..... | 37 |
| 7. References | 38 |

Table of Figures

| | |
|--|----|
| Figure 1 – CampusAssist’s interface | 3 |
| Figure 2 - Class Diagram for CampusAssist Web Application | 6 |
| Figure 3 - Use Case Diagram for CampusAssist..... | 7 |
| Figure 4 - Sequence Diagram – Book Appointment Flow | 8 |
| Figure 5 - Robustness Diagram – Submit Feedback Flow | 9 |
| Figure 6 - Adapted Java Model Classes in Web Project..... | 10 |
| Figure 7 - Student Login Interface | 11 |
| Figure 8 - SQLite view of the students IDs and passwords | 11 |
| Figure 9 -Student s1002 login | 12 |
| Figure 10 - Menu for 1002 student | 12 |
| Figure 11 - Student Dashboard Book Appointment button..... | 13 |
| Figure 12 - Booking of a financial appointment | 13 |
| Figure 13 - Financial Meeting Added..... | 14 |
| Figure 14 - Appointment Record in SQLite | 14 |
| Figure 15 - Feedback Submission Interface | 15 |
| Figure 16 - Feedback Insertion Logic via SubmitFeedbackServlet | 15 |
| Figure 17 - Student View Feedback Page – Displaying Feedback by Support Type, Date, and Time with a Navigation Option to Return to Dashboard | 16 |
| Figure 18 - Accessing The Support Services | 16 |
| Figure 19 - FAQ Menu..... | 17 |
| Figure 20 - Admin Login Interface | 18 |
| Figure 21 - SQLite Admin ID & password | 18 |
| Figure 22 - Admin Dashboard | 19 |
| Figure 23 - Admin Dashboard JSP Code Structure | 19 |
| Figure 24 - All Student Feedback..... | 20 |
| Figure 25 - Manage Feedback Button | 20 |
| Figure 26 - Categorise feedback and answer from admin to student | 20 |
| Figure 27 - Feedback Category Updated | 21 |
| Figure 28 - View Feedback Chart button | 21 |
| Figure 29 - Admin Feedback Category Chart | 22 |
| Figure 30- Admin FAQ Management Panel | 23 |
| Figure 31 - Admin Analytics Dashboard..... | 24 |
| Figure 32- Appointment Pending Status and Action Options..... | 25 |
| Figure 33 - Appointment After Admin Approval | 26 |
| Figure 34 - Feedback Submission Servlet (SubmitFeedbackServlet.java)..... | 28 |
| Figure 35 - Feedback Form JSP Page (giveFeedback.jsp)..... | 28 |
| Figure 36 - View Component (studentDashboard.jsp) | 29 |
| Figure 37 - Controller Component (AppointmentActionServlet.java)..... | 30 |
| Figure 38 - Model Layer (appointments table in SQLite)..... | 30 |
| Figure 39 - Input Validation and Error Handling in BookAppointmentServlet | 31 |
| Figure 40 - Student Booking an Appointment (Mental Health) | 33 |
| Figure 41 - Dashboard Displaying Student’s Booking | 33 |
| Figure 42 - Admin Cancelling the Appointment (Test Scenario) | 34 |
| Figure 43 - Submitfeedback fix..... | 35 |

1. Introduction

CampusAssist is an online student support tool created as a component of the Object Oriented Design and Development (OODD) course at Solent University. The system builds upon an earlier group console application from AE1, evolving it into a comprehensive JSP web application featuring persistent storage and distinct roles for both students and administrators. This project aims to showcase the concepts of object-oriented programming, modular architecture, and separation of concerns by developing a practical student service platform. The system offers features for scheduling appointments, providing and overseeing feedback, utilizing support services, and conducting analysis on student engagement data.

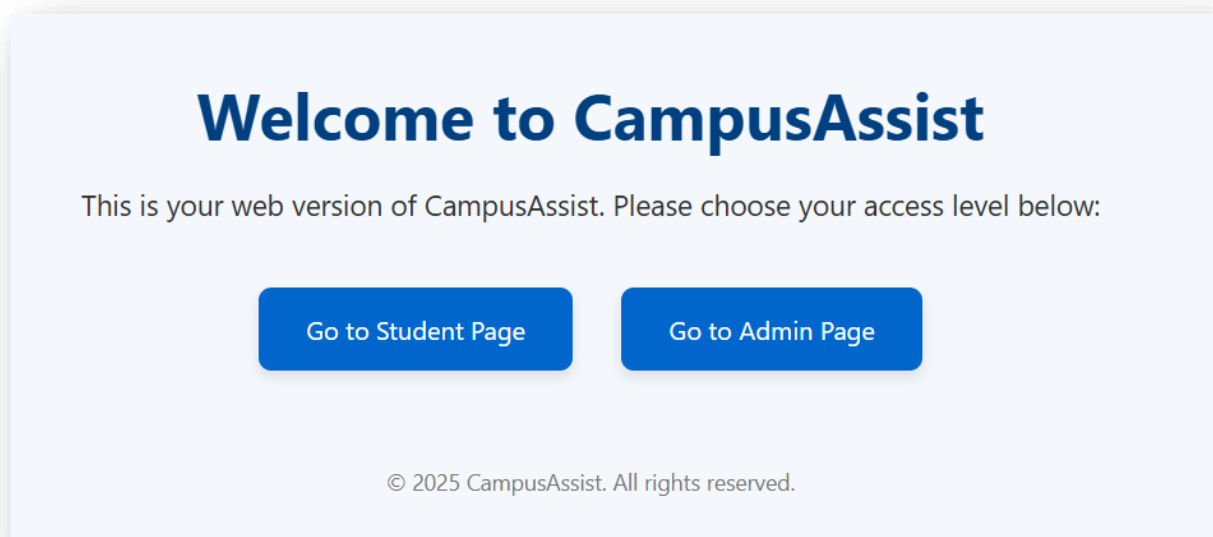


Figure 1 – CampusAssist's interface

1.1. Project objective

The purpose of this personal assignment is to modify and enhance the group-created CampusAssist project into a fully operational web application that fulfills the AE2 requirements. The main goals encompass:

- Transferring the fundamental logic from the AE1 console app to a web-oriented structure utilizing JSP and JDBC;

- Creating distinct access and interfaces for students and administrators, incorporating session-based login systems;
- Activating essential functionalities like: Scheduling and managing appointments; Submission of feedback, classification, and reply; Access to FAQs and administration of CRUD management; Analytics for visual feedback and searchable support material;
- Exhibiting analysis and design based on objects through class, sequence, and use case diagrams;
- Creating a comprehensive technical document to outline the design procedure, implementation specifics, and testing results;

This initiative replicates a practical university support framework featuring various modules, user roles, and backend functionality, emphasizing maintainability, scalability, and access control based on roles.

1.2. Used technologies

| Technology | Purpose |
|----------------------|--|
| Java (JDK 21) | Essential application functionality (models, management, server-side processing) |
| JSP | Generation of frontend views for student and admin interfaces |
| Servlets | Layer responsible for managing requests and business logic |
| JDBC | Database connection and functions |
| SQLite | Ongoing local storage for bookings, reviews, frequently asked questions |
| Apache Tomcat | Server for deploying and testing the application |
| Draw.io | The creation of the diagrams (UML: use case, sequence, robustness) |
| JUnit | Unit testing for Java classes in the backend |
| CSS | Fundamental design enhancements for a better user experience |
| NetBeans IDE | Environment for coding and debugging integration |

These technologies were chosen to match module needs and to represent a practical tech stack frequently utilized in full-stack web development projects.

2. Architecture of the System and Design of Classes:

This part describes the structural design of the CampusAssist system. It describes the application of object-oriented principles to develop a modular, scalable web application and details the architectural transformation from the initial AE1 console version to a web version for AE2.

2.1. Class Structure Overview:

CampusAssist is created utilizing an object-oriented methodology, where duties are divided into model, manager, and control (servlet) tiers. This design guarantees adaptability, reusability, and simplified testing.

Classes and their roles:

| Class Name | Role |
|---------------------------|--|
| Student | Denotes student users (ID, name, login information) |
| Admin | Denotes administrative users with heightened privileges |
| Appointment | Data structure for a scheduled support session |
| AppointmentManager | Manages session scheduling, conflict resolution, and time organization |
| Feedback | Stores feedback, review, and links to a booking |
| FeedbackManager | Collects, saves, and organizes feedback |
| ReminderManager | Creates reminders according to current appointments |
| FAQ | Template for common inquiries overseen by the administrator |
| Analytics | Compiles feedback and session information for reports |

Every one of these classes adheres to the Single Responsibility Principle, promoting a distinct separation of logic and enhancing reusability for both the console and web versions.

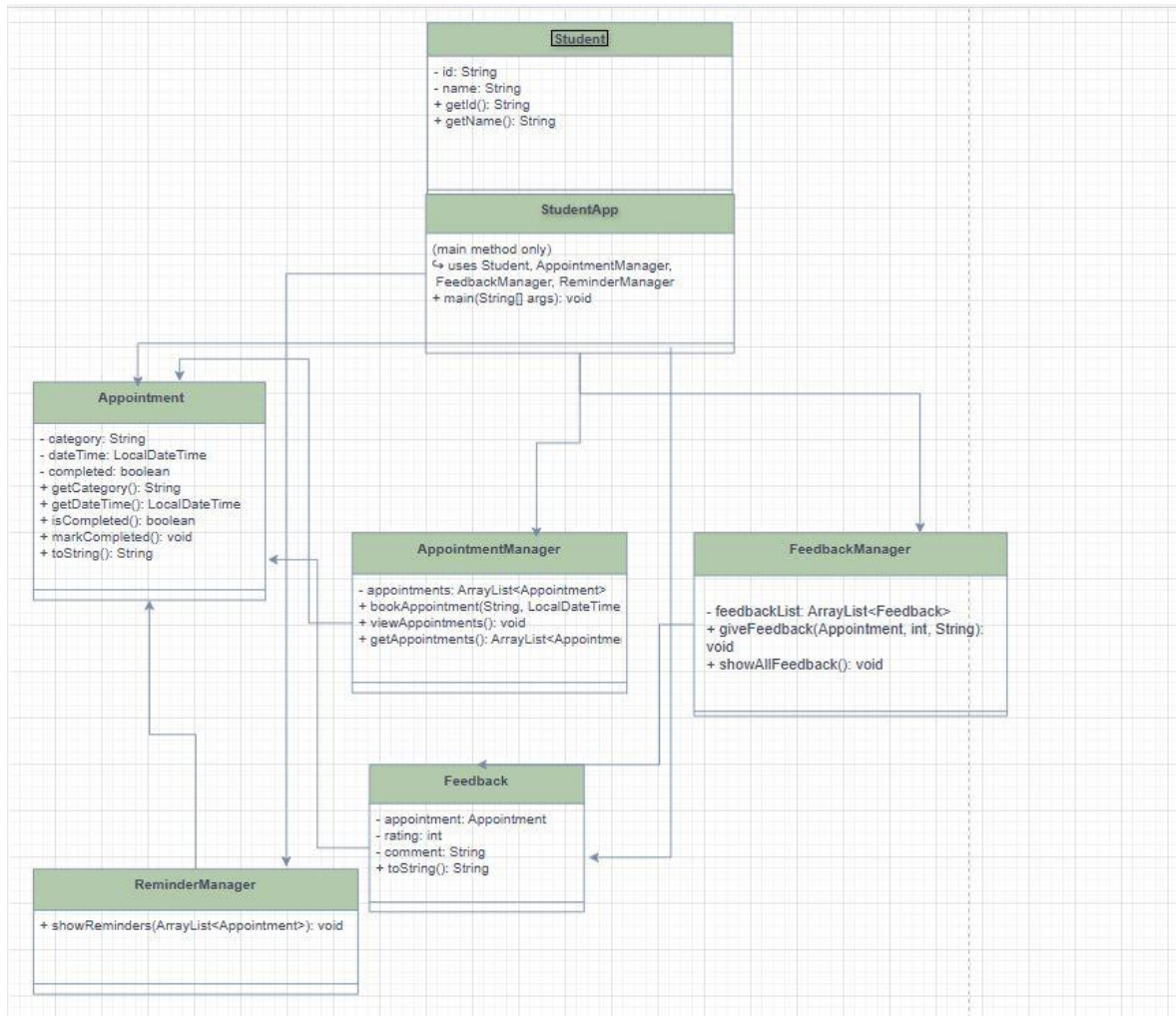


Figure 2 - Class Diagram for CampusAssist Web Application

2.2. Use Case Diagram

The system accommodates two main roles: Student and Admin. Each engages with a unique array of characteristics:

Students are able to:

- Log in;
- Book appointments;
- View history;
- Submit feedback;
- Browse support service and FAQ;

Admins are able to:

- View, approve, reschedule, or even cancel appointments;
- Categorize and respond to feedback;
- Manage the FAQ;

- View analytics and filter feedback reports;

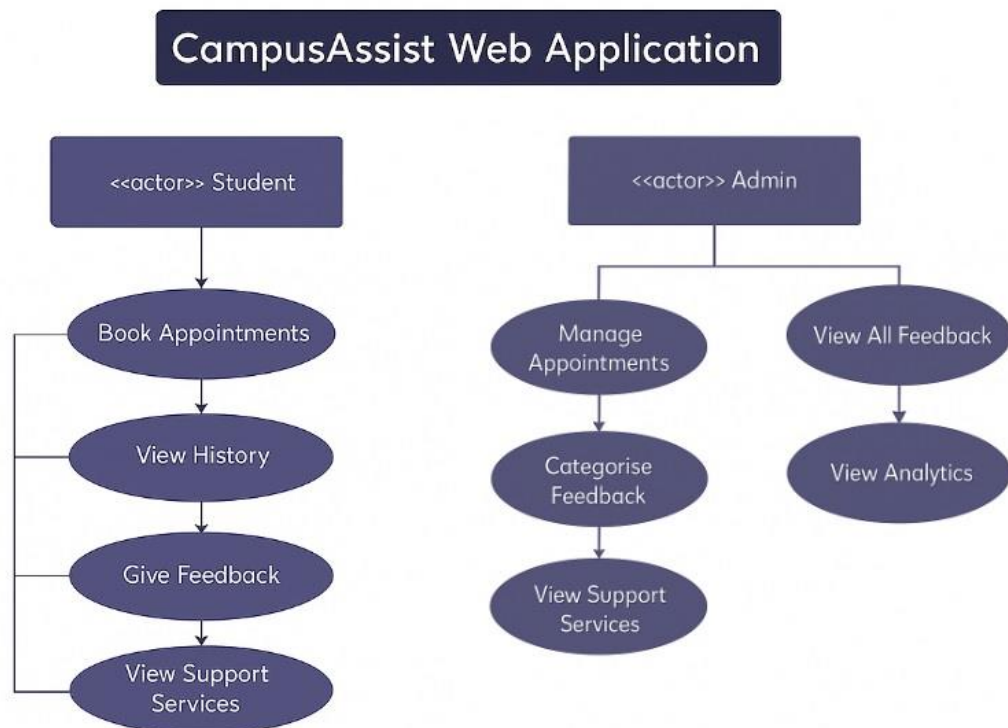


Figure 3 - Use Case Diagram for CampusAssist

2.3. The sequence diagram:

This diagram demonstrates the connection among student input, JSP page, servlet, logic layer, and the database during a student's booking of a support appointment:

Participants:

- Student(actor);
- Bookappointments.jsp;
- BookappointmentServlet;
- AppointmentManager;
- SQLite DB;

Sequence Diagram – Book Appointment Flow

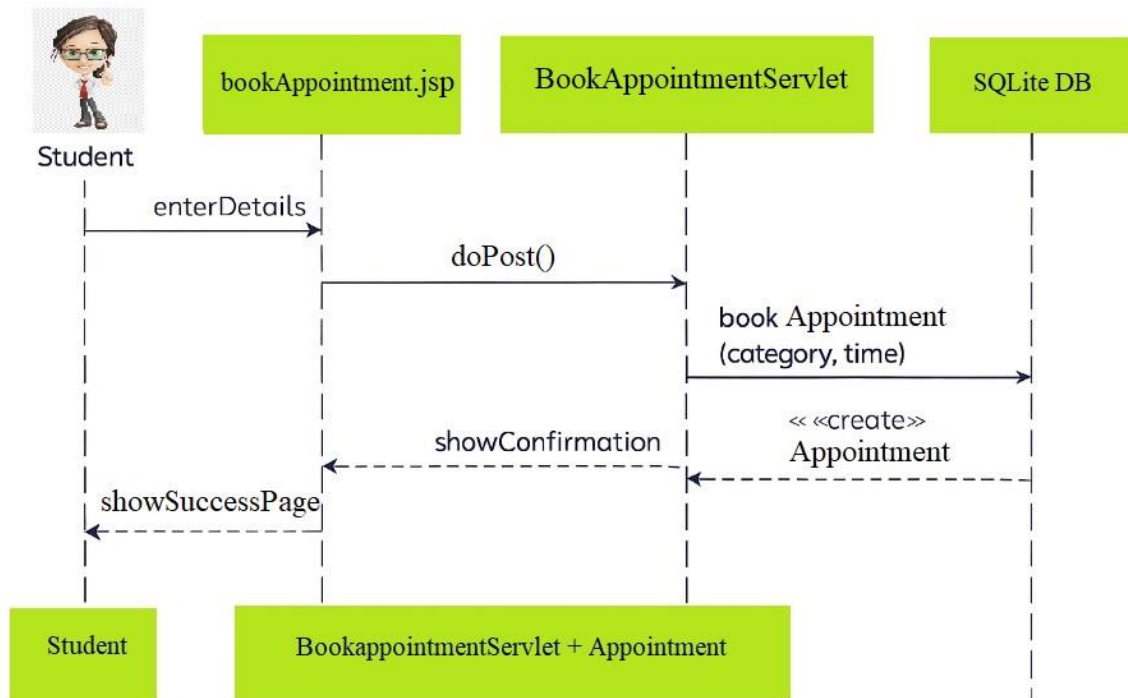


Figure 4 - Sequence Diagram – Book Appointment Flow

2.4. The robustness diagram:

- Boundary: givefeedback.jsp;
- Control: Submitfeedback Servlet;
- Entity: FeedbackManager + Feedback.java

It demonstrates how the system upholds clear role separation while facilitating the transfer of data from UI to logic to storage.

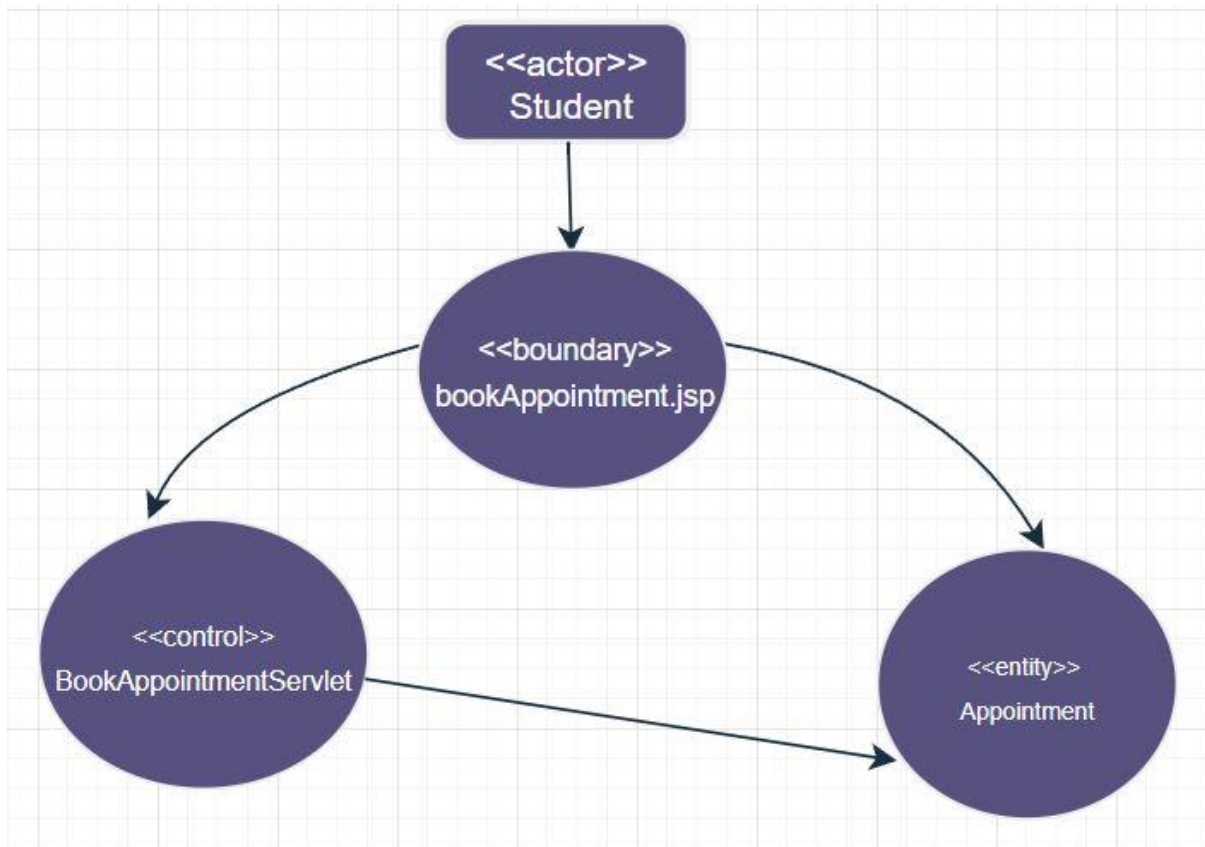


Figure 5 - Robustness Diagram – Submit Feedback Flow

3. Execution and Functionality Advancement

This section outlines the key features created during AE2, emphasizing both the modified logic from the AE1 group project and the newly added web functionalities. The project was reconstructed in NetBeans as a JSP web application and evaluated with Apache Tomcat 10.1 and an SQLite database.

3.1. Overview of Adapted Code from Group Project

The core object-oriented code from AE1 was retained and transferred to the web setting. Core Java classes like Student, Appointment, Feedback, FAQ, and their related manager classes (AppointmentManager, FeedbackManager, etc.) were utilized without changing their functionality.

The initial group project included console input and output logic, which was eliminated, and a new controller layer was introduced via servlets (e.g., BookAppointmentServlet,

SubmitFeedbackServlet) to manage web interactions. These servlets communicate with the original manager classes and interact with JSP pages to display the outcomes.

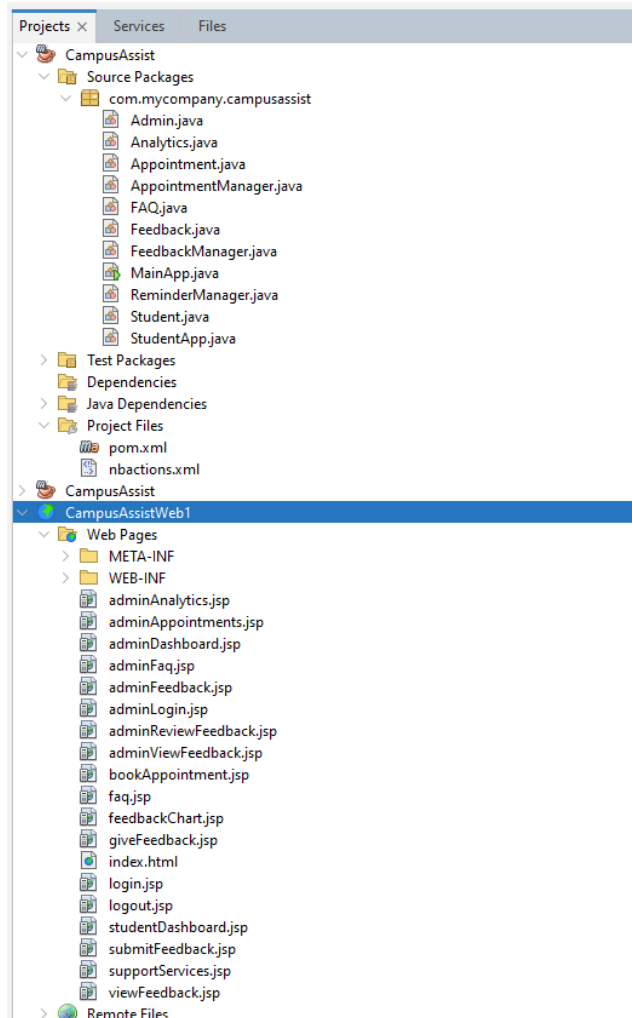


Figure 6 - Adapted Java Model Classes in Web Project

3.2. Student's perspective Functionality

Login and Session Control

Students log into the system through login.jsp by entering an ID and password, which are currently authenticated using hardcoded credentials or database entries. Upon a successful outcome, sessions are started and users are directed to studentDashboard.jsp.

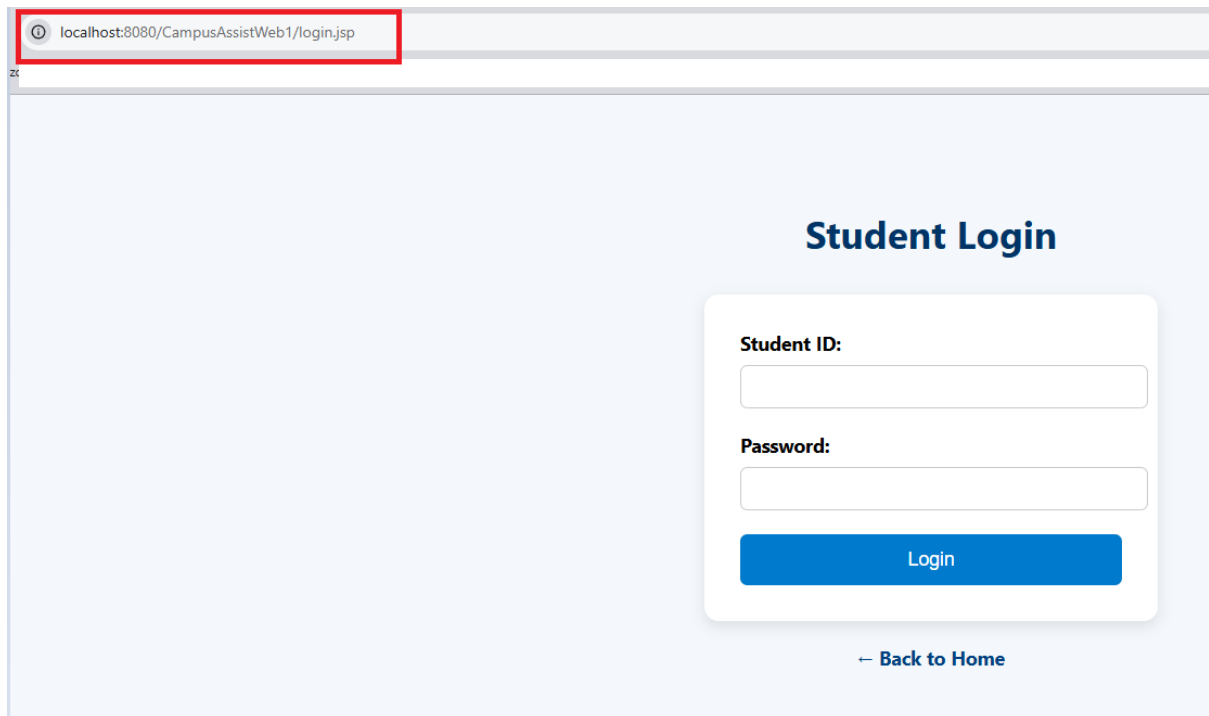


Figure 7 - Student Login Interface

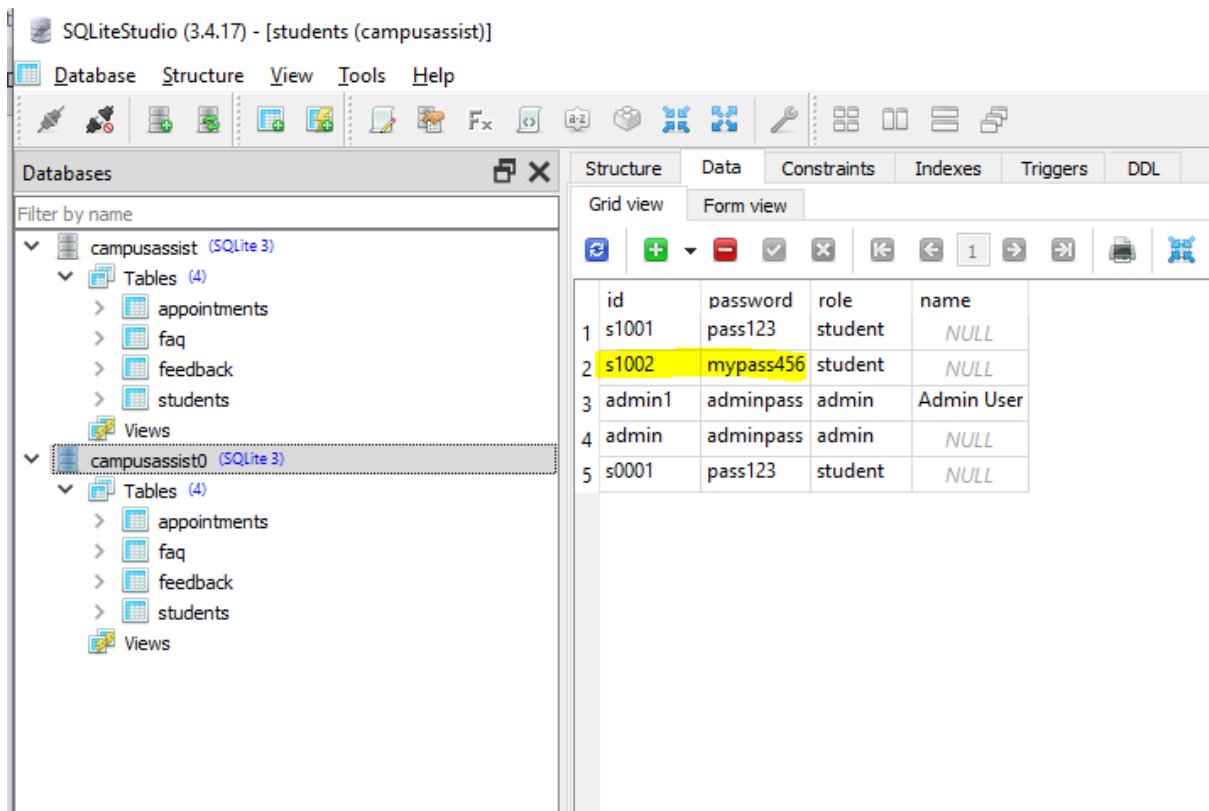


Figure 8 - SQLite view of the students IDs and passwords

We can take for instance the first student and try and get into the menu:

Student Login

Student ID:

Password:

Login

← Back to Home

Figure 9 -Student s1002 login

Welcome, Student s1002!

Your Appointments:

| Type | Date | Time | Details |
|---------------|------------|-------|--|
| Mental Health | 2025-04-23 | 18:15 | Insomnia. |
| Mental Health | 2025-04-25 | 03:14 | Anxiety. |
| Mental Health | 2025-04-23 | 06:15 | Insomnia. |
| Academic | 2025-04-27 | 09:00 | Hello, I have some questions about the new scholar year. Could you please assist? Thank you! |
| Financial | 2025-05-16 | 10:30 | Hello, Please support with the council tax student reduction letter and how to get in possession of it. Thank you! |
| Academic | 2025-06-13 | 13:00 | I need help my disertation. |

Book a New Appointment

Give Feedback

View Feedback

Browse Support Services

View FAQs

Logout

← Back to Home

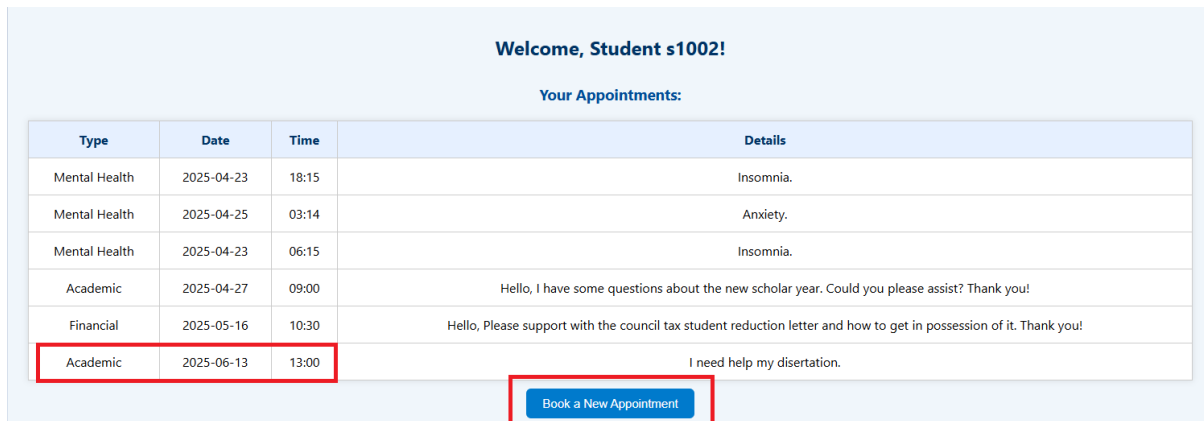
Figure 10 - Menu for 1002 student

As you can see in Figure 9 we can see all of the appointments that student 1002 has had. I have added these for a better understading and to better illustrate the functionality. This is how the student dashboard looks like.

Booking and Viewing Appointments

Students are able to schedule a new appointment through `bookAppointment.jsp`. The `BookAppointmentServlet` handles the form and stores information in the database. Scheduled appointments are shown dynamically on `studentDashboard.jsp`.

I will continue using student 1002 for the next example and I will try to book an appointment for the 14th of June just to differentiate as the last one is on 13th on June.



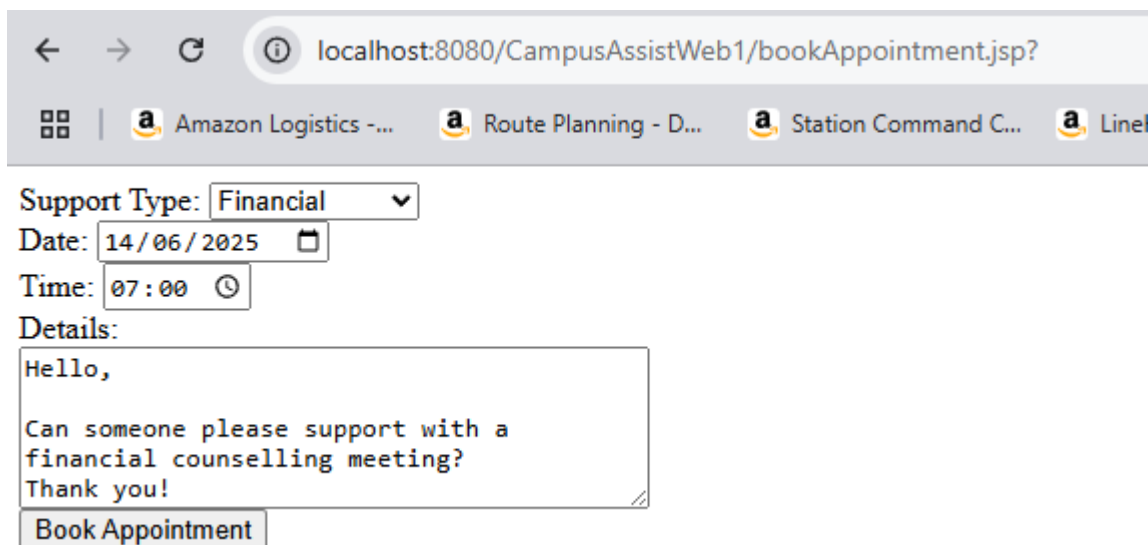
The screenshot shows a web interface for a student dashboard. At the top, it says "Welcome, Student s1002!". Below that, there's a section titled "Your Appointments:" which contains a table of appointments. The table has four columns: Type, Date, Time, and Details. The appointments listed are:

| Type | Date | Time | Details |
|---------------|------------|-------|--|
| Mental Health | 2025-04-23 | 18:15 | Insomnia. |
| Mental Health | 2025-04-25 | 03:14 | Anxiety. |
| Mental Health | 2025-04-23 | 06:15 | Insomnia. |
| Academic | 2025-04-27 | 09:00 | Hello, I have some questions about the new scholar year. Could you please assist? Thank you! |
| Financial | 2025-05-16 | 10:30 | Hello, Please support with the council tax student reduction letter and how to get in possession of it. Thank you! |
| Academic | 2025-06-13 | 13:00 | I need help my disertation. |

Below the table, there is a blue button labeled "Book a New Appointment".

Figure 11 - Student Dashboard Book Appointment button

Next we will see how the booking option looks like, and I will try to book a financial meeting for instance just for the sake of functionality.



The screenshot shows a web browser window with the URL `localhost:8080/CampusAssistWeb1/bookAppointment.jsp?`. The browser's address bar shows several tabs: "Amazon Logistics -...", "Route Planning - D...", "Station Command C...", and "Line...". The form itself has the following fields:

- Support Type:** A dropdown menu with "Financial" selected.
- Date:** A date picker showing "14/06/2025".
- Time:** A time picker showing "07:00".
- Details:** A text area containing the message: "Hello, Can someone please support with a financial counselling meeting? Thank you!".
- Book Appointment:** A button at the bottom of the form.

Figure 12 - Booking of a financial appointment

I have also written a message in order to identify my last appointment easier.

Welcome, Student s1002!

Your Appointments:

| Type | Date | Time | Details |
|---------------|------------|-------|--|
| Mental Health | 2025-04-23 | 18:15 | Insomnia. |
| Mental Health | 2025-04-25 | 03:14 | Anxiety. |
| Mental Health | 2025-04-23 | 06:15 | Insomnia. |
| Academic | 2025-04-27 | 09:00 | Hello, I have some questions about the new scholar year. Could you please assist? Thank you! |
| Financial | 2025-05-16 | 10:30 | Hello, Please support with the council tax student reduction letter and how to get in possession of it. Thank you! |
| Academic | 2025-06-13 | 13:00 | I need help my disertation. |
| Financial | 2025-06-14 | 07:00 | Hello, Can someone please support with a financial counselling meeting? Thank you! |

[Book a New Appointment](#)
[Give Feedback](#)
[View Feedback](#)
[Browse Support Services](#)
[View FAQs](#)
[Logout](#)

Figure 13 - Financial Meeting Added

SQLiteStudio (3.4.17) - [appointments (campusassist)]

Database Structure View Tools Help

Databases

- campusassist (SQLite 3)
 - Tables (4)
 - appointments
 - faq
 - feedback
 - students
 - Views
- campusassist0 (SQLite 3)
 - Tables (4)
 - appointments
 - faq
 - feedback
 - students
 - Views

Grid view Form view

Filter data Total rows loaded: 8

| id | studentID | type | date | time | details | status |
|----|-----------|---------------|------------|-------|---|-------------|
| 1 | s1002 | Mental Health | 2025-04-23 | 18:15 | Insomnia. | Approved |
| 2 | s1002 | Mental Health | 2025-04-25 | 03:14 | Anxiety. | Cancelled |
| 3 | s1002 | Mental Health | 2025-04-23 | 06:15 | Insomnia. | Rescheduled |
| 4 | s1002 | Academic | 2025-04-27 | 09:00 | Hello,... | Approved |
| 5 | s1002 | Financial | 2025-05-16 | 10:30 | Hello,... | Approved |
| 6 | s0001 | Academic | 2025-04-26 | 07:00 | Hello, please let me know when you have a financial advisor available. Thank you! | Approved |
| 7 | s1002 | Academic | 2025-06-13 | 13:00 | I need help my disertation. | NULL |
| 8 | s1002 | Financial | 2025-06-14 | 07:00 | Hello,... | NULL |

Figure 14 - Appointment Record in SQLite

Of course for a better verification we also have to check the SQLite database and we can see that the financial booking I have added has gone through.

Feedback submission

Following an appointment, students can provide feedback via giveFeedback.jsp. The system associates each feedback with a particular appointment and stores it using SubmitFeedbackServlet:

← → ↻ ⓘ localhost:8080/CampusAssistWeb1/giveFeedback.jsp?

Amazon Logistics - ... Route Planning - D... Station Command C... LineHaul Agg

Give Feedback for Your Appointments

Select Appointment: Financial on 2025-06-14 at 07:00 ▼

Feedback:

Thank you for considering my meeting!

Submit Feedback

[Back to Dashboard](#)

Figure 15 - Feedback Submission Interface

I have focused on the same booking I used previously to follow the functionality.

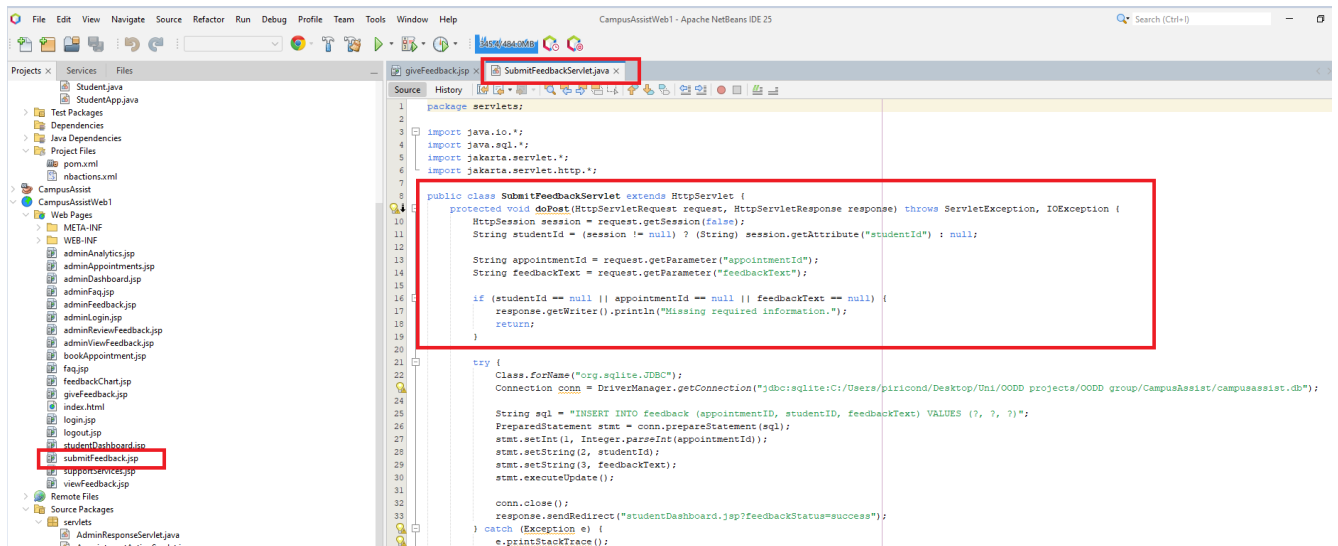


Figure 16 - Feedback Insertion Logic via SubmitFeedbackServlet

This servlet handles feedback provided by students. It verifies input, links to the SQLite database, and saves the feedback. The process ends with a redirect to the student dashboard after a successful operation

And in order to check if it went through we can go back to the menu and select View Feedback and it should show us all the feedbacks we have given so far.

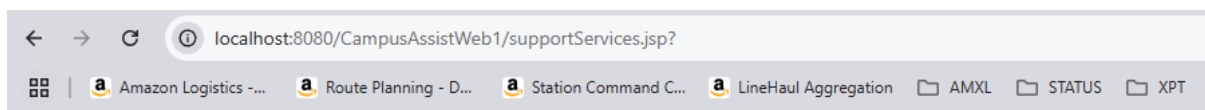
| Feedback for Your Appointments | | | |
|--------------------------------|------------|-------|--|
| Support Type | Date | Time | Feedback |
| Mental Health | 2025-04-23 | 18:15 | Thanks :) |
| Mental Health | 2025-04-23 | 18:15 | Thanks :) |
| Mental Health | 2025-04-23 | 18:15 | 5 stars, really good counselling! |
| Mental Health | 2025-04-23 | 18:15 | Keep up the good work! ^^ |
| Mental Health | 2025-04-23 | 18:15 | thank you! |
| Mental Health | 2025-04-23 | 18:15 | test |
| Mental Health | 2025-04-23 | 18:15 | test2 |
| Mental Health | 2025-04-23 | 18:15 | Feedback test |
| Academic | 2025-04-27 | 09:00 | Thank you very much for approving my appointment! ^^ |
| Academic | 2025-06-13 | 13:00 | Booked at 13:00. Happy to help! |
| Financial | 2025-06-14 | 07:00 | Thank you for considering my meeting! |

[Back to Dashboard](#)

Figure 17 - Student View Feedback Page – Displaying Feedback by Support Type, Date, and Time with a Navigation Option to Return to Dashboard

FAQ and Support Services Access

Students can access FAQs (faq.jsp) and support details (supportServices.jsp). These pages are view-only and assist users in navigating available resources.



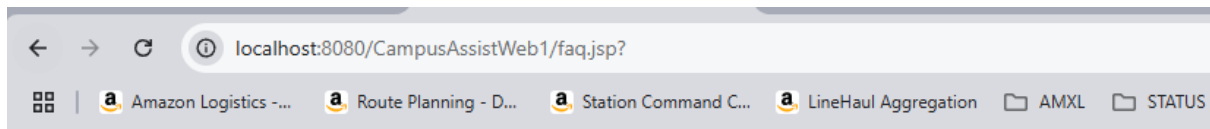
Explore Our Support Services

- Mental Health Support**
 Our counselors are here to listen and help you manage anxiety, stress, and other personal challenges.
- Academic Support**
 Get help with time management, study strategies, and course-specific tutoring.
- Financial Support**
 We offer guidance on managing student finances and can connect you to emergency funding options.

[← Back to Dashboard](#)

Figure 18 - Accessing The Support Services

When we access the Support Services we can see the options we have in booking via the Student platform in case the students are wondering weather their questions can be answered, and as an extra we also have the FAQ if the things are still unclear or in case anyone needs anymore details.



Frequently Asked Questions

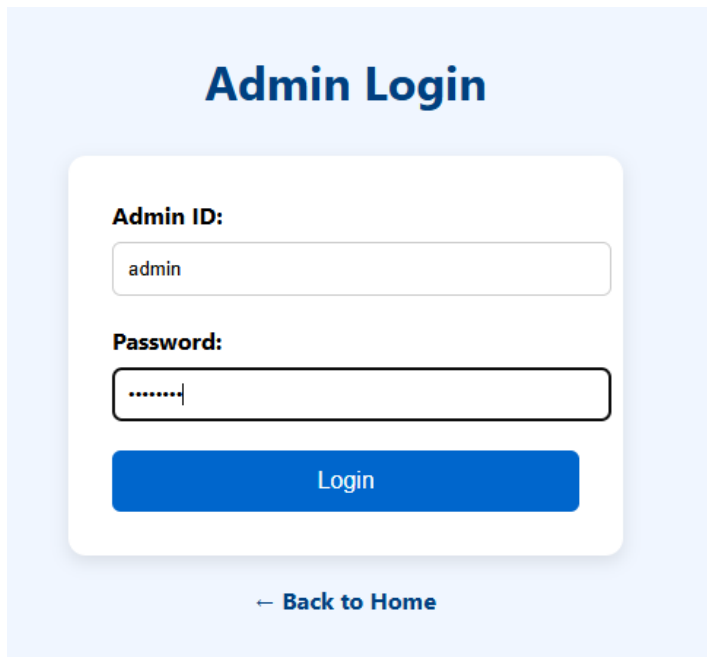
- **How do I book an appointment?**
Go to the student dashboard and click "Book a New Appointment".
- **Can I cancel my appointment?**
Please contact support to cancel appointments.
- **What types of support are available?**
Mental Health, Academic, and Financial.
- **How long is each support session?**
Each session typically lasts 30 to 45 minutes.
- **Can I reschedule my appointment?**
Yes, but you must request changes at least 24 hours in advance.
- **Is the support confidential?**
Yes. All student support sessions are strictly confidential.
- **How many appointments are financially supported by the University?**
The first 10 appointments.

[← Back to Dashboard](#)

Figure 19 - FAQ Menu

In the FAQ Menu from the Student we have a number of 7 questions that have been asked more often.

3.3. Admin's perspective functionality



The image shows a web interface for Admin Login. It has a light blue background. At the top, the text "Admin Login" is displayed in a large, bold, dark blue font. Below this, there is a white rounded rectangle containing the login form. The form has two input fields: "Admin ID:" with the text "admin" entered, and "Password:" with masked characters ".....". Below the password field is a blue button with the text "Login". At the bottom of the white rectangle, there is a link that says "← Back to Home".

Figure 20 - Admin Login Interface

We can proceed as we did for the student and take the passwords from the database.

| | id | password | role | name |
|---|--------|-----------|---------|------------|
| 1 | s1001 | pass123 | student | NULL |
| 2 | s1002 | mypass456 | student | NULL |
| 3 | admin1 | adminpass | admin | Admin User |
| 4 | admin | adminpass | admin | NULL |
| 5 | s0001 | pass123 | student | NULL |

Figure 21 - SQLite Admin ID & password

Upon entering the ID admin and the password admin pass we will be directed to the admin dashboard which has an admin menu:

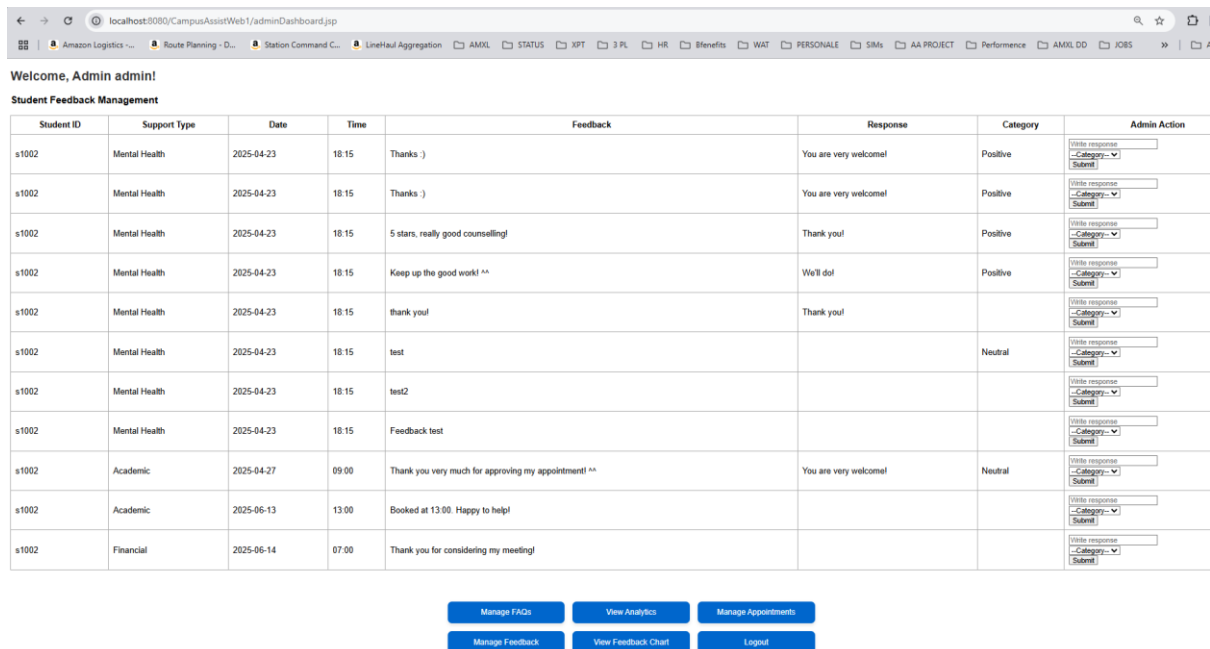


Figure 22 - Admin Dashboard

Admins are able to view, approve, cancel, or reschedule appointments through `adminAppointments.jsp`. The admin dashboard shows a status column featuring color-coded signals.

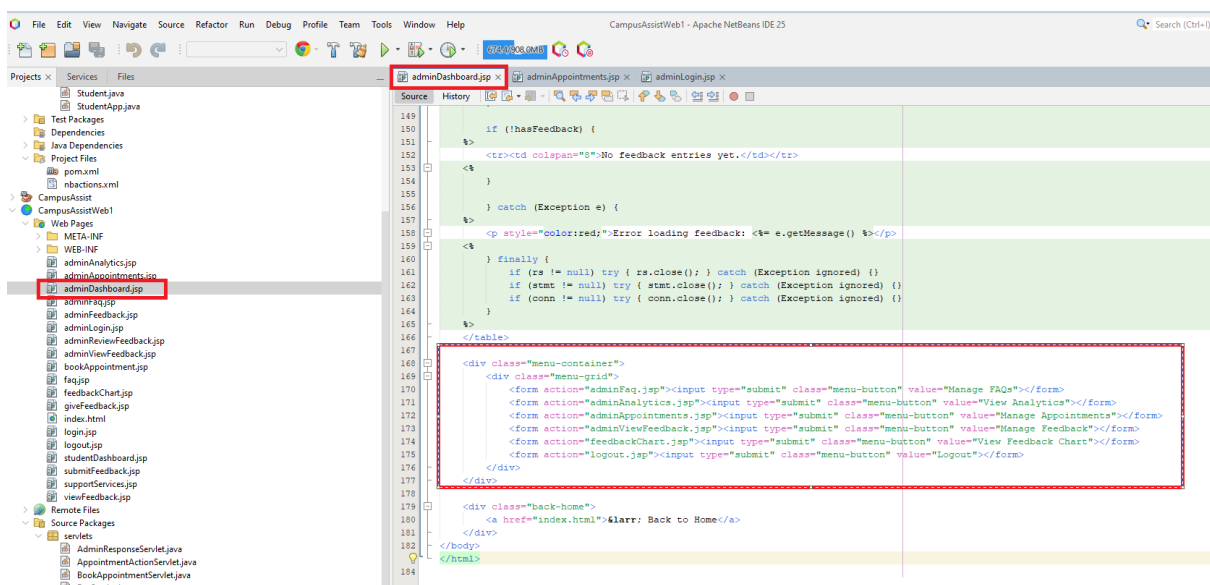
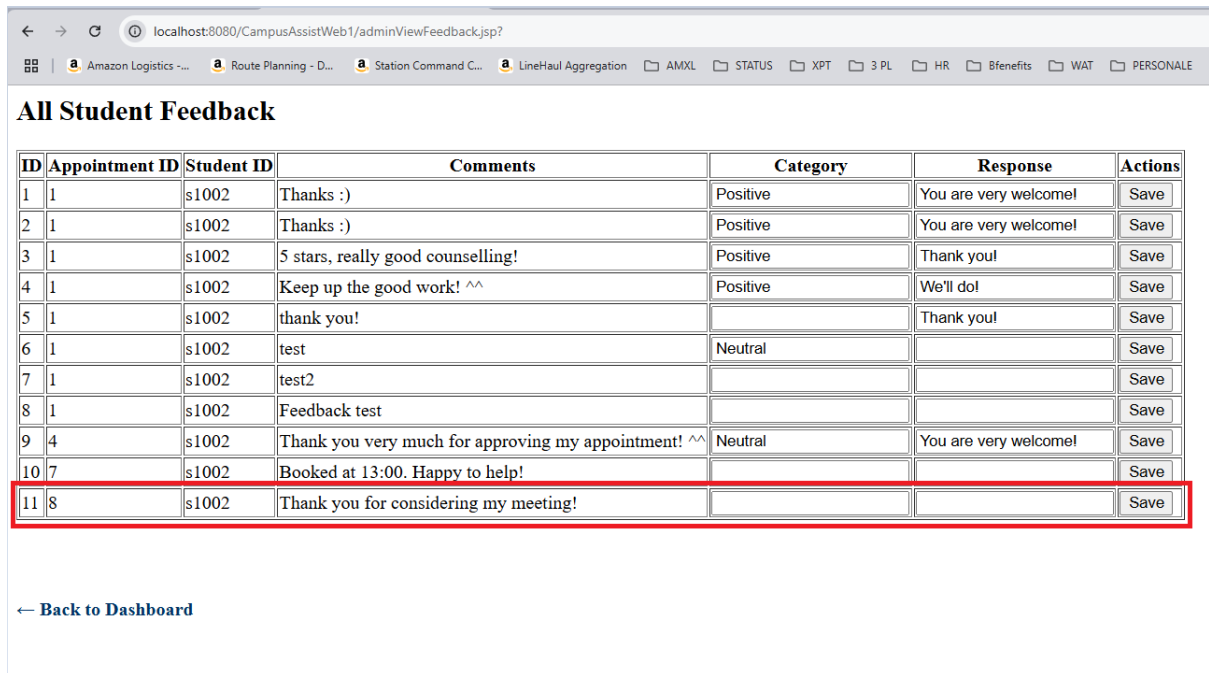


Figure 23 - Admin Dashboard JSP Code Structure

This screenshot illustrates the layout of `adminDashboard.jsp`, which offers navigation for all admin features. It features error management for obtaining feedback data and a grid-format admin menu that connects to key modules: FAQs, Analytics, Appointment Management, Feedback Review, Feedback Chart, and Logout. This structured arrangement guarantees fast access to all tools needed for administrative supervision.

Feedback categorisation and analytics

Admins can assess all feedback in adminViewFeedback.jsp, categorize it (e.g., Positive, Neutral), and provide a response. The analytics dashboard (adminAnalytics.jsp) displays summary tables.



| ID | Appointment ID | Student ID | Comments | Category | Response | Actions |
|----|----------------|------------|--|----------|-----------------------|---------|
| 1 | 1 | s1002 | Thanks :) | Positive | You are very welcome! | Save |
| 2 | 1 | s1002 | Thanks :) | Positive | You are very welcome! | Save |
| 3 | 1 | s1002 | 5 stars, really good counselling! | Positive | Thank you! | Save |
| 4 | 1 | s1002 | Keep up the good work! ^^ | Positive | We'll do! | Save |
| 5 | 1 | s1002 | thank you! | | Thank you! | Save |
| 6 | 1 | s1002 | test | Neutral | | Save |
| 7 | 1 | s1002 | test2 | | | Save |
| 8 | 1 | s1002 | Feedback test | | | Save |
| 9 | 4 | s1002 | Thank you very much for approving my appointment! ^^ | Neutral | You are very welcome! | Save |
| 10 | 7 | s1002 | Booked at 13:00. Happy to help! | | | Save |
| 11 | 8 | s1002 | Thank you for considering my meeting! | | | Save |

[← Back to Dashboard](#)

Figure 24 - All Student Feedback

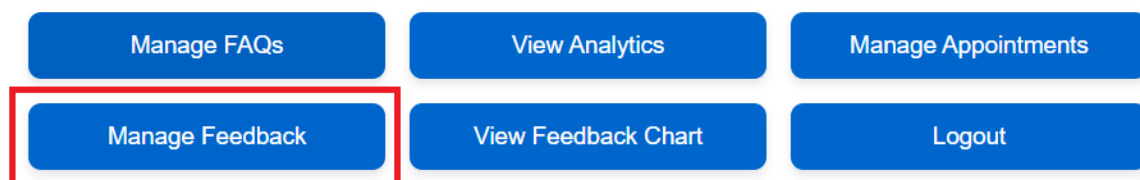
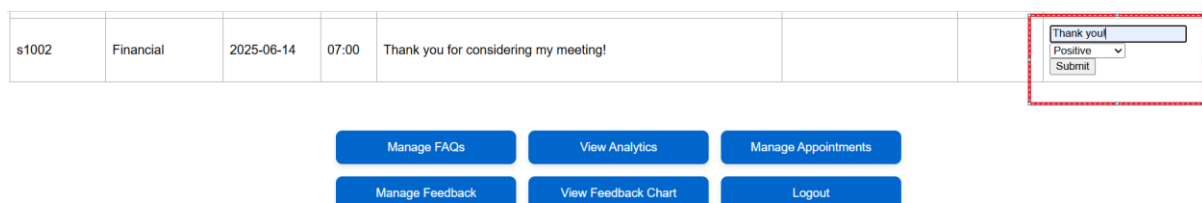


Figure 25 - Manage Feedback Button

As soon as we enter the Manage Feedback button we can see the feedback I have written previously, the one from the financial booking. We can also categorize the feedback as positive, negative or neutral:



| | | | | | | |
|-------|-----------|------------|-------|---------------------------------------|--|--|
| s1002 | Financial | 2025-06-14 | 07:00 | Thank you for considering my meeting! | | |
|-------|-----------|------------|-------|---------------------------------------|--|--|

Manage FAQs

View Analytics

Manage Appointments

Manage Feedback

View Feedback Chart

Logout

Figure 26 - Categorise feedback and answer from admin to student

As soon as we save the category of the feedback and send a message or not, the new status will appear.

All Student Feedback

| ID | Appointment ID | Student ID | Comments | Category | Response | Actions |
|----|----------------|------------|--|----------|-----------------------|---------|
| 1 | 1 | s1002 | Thanks :) | Positive | You are very welcome! | Save |
| 2 | 1 | s1002 | Thanks :) | Positive | You are very welcome! | Save |
| 3 | 1 | s1002 | 5 stars, really good counselling! | Positive | Thank you! | Save |
| 4 | 1 | s1002 | Keep up the good work! ^^ | Positive | We'll do! | Save |
| 5 | 1 | s1002 | thank you! | | Thank you! | Save |
| 6 | 1 | s1002 | test | Neutral | | Save |
| 7 | 1 | s1002 | test2 | | | Save |
| 8 | 1 | s1002 | Feedback test | | | Save |
| 9 | 4 | s1002 | Thank you very much for approving my appointment! ^^ | Neutral | You are very welcome! | Save |
| 10 | 7 | s1002 | Booked at 13:00. Happy to help! | | | Save |
| 11 | 8 | s1002 | Thank you for considering my meeting! | Positive | Thank you! | Save |

[← Back to Dashboard](#)

Figure 27 - Feedback Category Updated

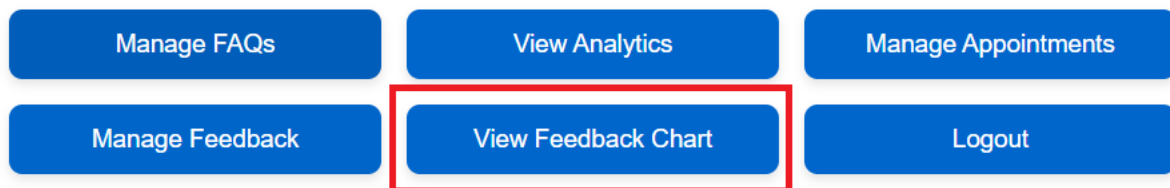


Figure 28 - View Feedback Chart button

Now from the admin menu we can go to feedback chart and have an ampler view.

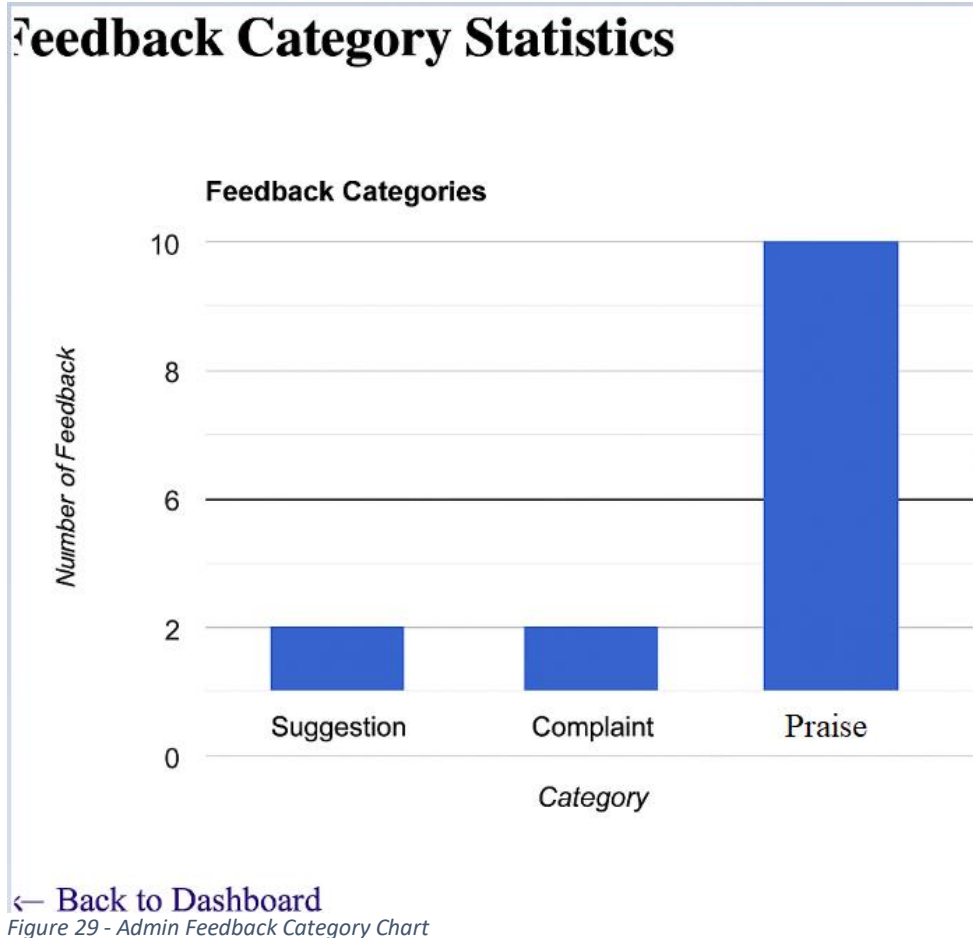
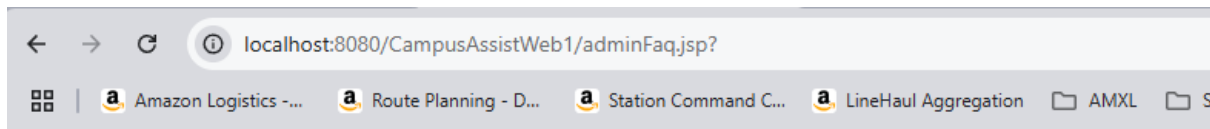


Figure 29 - Admin Feedback Category Chart

This graphical chart shows the variety of feedback types classified by the administrator. It aids in recognizing patterns in user satisfaction by indicating how many entries belong to categories like Suggestion, Complaint, or Praise. In this instance, the majority of feedback is categorized as “Praise,” reflecting a favorable attitude from students.

FAQ management

Admins utilize `adminFaq.jsp` to insert, modify, or remove FAQs from the database.



Manage FAQs

Question:

Answer:

| ID | Question | Answer | Actions | |
|----|------------------------------|------------------------------|---------------------------------------|---------------------------------------|
| 1 | How do I book an appointm | Go to the student dashboar | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 2 | Can I cancel my appointme | Please contact support to c | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 3 | What types of support are a | Mental Health, Academic, a | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 4 | How long is each support s | Each session typically lasts | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 5 | Can I reschedule my appoi | Yes, but you must request c | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 6 | Is the support confidential? | Yes. All student support ses | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |
| 7 | How many appointments ar | The first 10 appointments. | <input type="button" value="Update"/> | <input type="button" value="Delete"/> |

[← Back to Dashboard](#)

Figure 30- Admin FAQ Management Panel

The FAQ section enables administrators to handle commonly asked questions. Admins have the ability to add new questions, modify current entries, or remove old ones. The interface presents the FAQs in a straightforward table format with editable sections, ensuring that upkeep is effective and user-friendly.

Search and reporting features

The administration interface allows for searching and filtering options (e.g., by category of feedback or status of appointments). Reports are created in dynamic HTML tables, enabling real-time engagement.

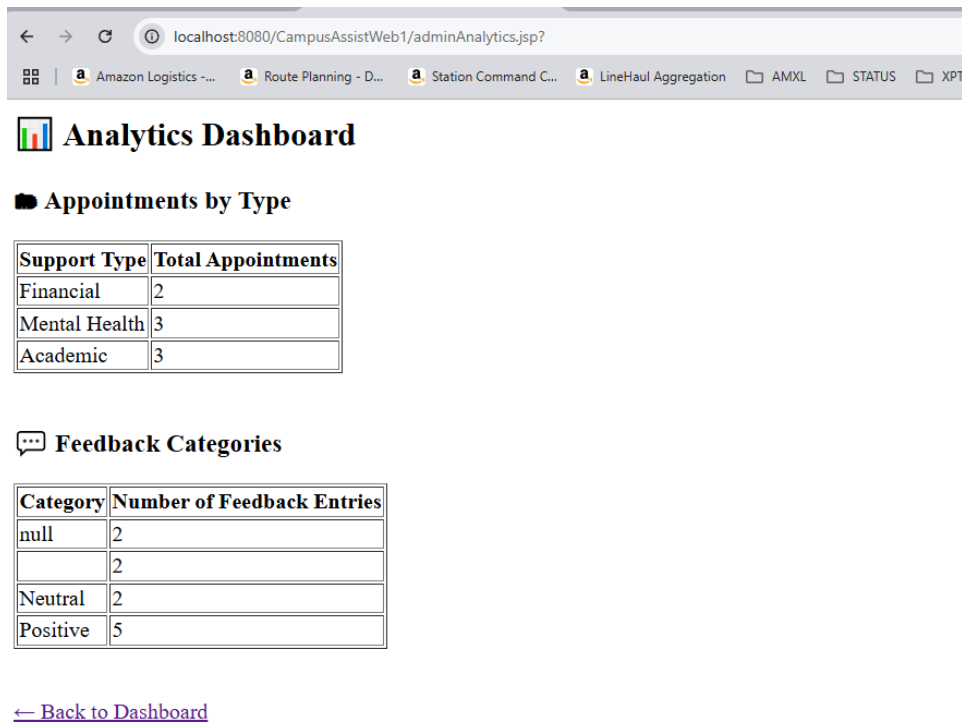


Figure 31 - Admin Analytics Dashboard

This dashboard offers a statistical overview of support utilization and responses. The initial section shows the count of appointments for each support type (Academic, Mental Health, Financial), whereas the subsequent section classifies feedback as Positive, Neutral, or uncategorized. These findings aid in making informed decisions by emphasizing service demand and user reactions.

Manage Appointments

The screenshot shows a web browser window with the URL `localhost:8080/CampusAssistWeb1/adminAppointments.jsp?`. The page title is "Admin - Manage Appointments". Below the title is a "Filter by Status:" dropdown menu set to "All". The main content is a table with 8 rows of appointment data. The 8th row is highlighted with a red dashed border. Below the table is a link "[← Back to Dashboard](#)".

| Appointment ID | Student ID | Type | Date | Time | Status | Actions |
|----------------|------------|---------------|------------|-------|-------------|---------------------------|
| 1 | s1002 | Mental Health | 23/04/2025 | 18:15 | Approved | Approve Reschedule Cancel |
| 2 | s1002 | Mental Health | 25/04/2025 | 03:14 | Cancelled | Approve Reschedule Cancel |
| 3 | s1002 | Mental Health | 23/04/2025 | 06:15 | Rescheduled | Approve Reschedule Cancel |
| 4 | s1002 | Academic | 27/04/2025 | 09:00 | Approved | Approve Reschedule Cancel |
| 5 | s1002 | Financial | 16/05/2025 | 10:30 | Approved | Approve Reschedule Cancel |
| 6 | s0001 | Academic | 26/04/2025 | 07:00 | Approved | Approve Reschedule Cancel |
| 7 | s1002 | Academic | 13/06/2025 | 13:00 | Pending | Approve Reschedule Cancel |
| 8 | s1002 | Financial | 14/06/2025 | 07:00 | Pending | Approve Reschedule Cancel |

Figure 32- Appointment Pending Status and Action Options

Figure 31 displays a Pending status appointment from the `adminAppointments.jsp` interface. It is the same appointment we have been following so far, which the admin hasn't handled yet. Admins have three action buttons available: Approve, Reschedule, or Cancel. This perspective assists administrators in effectively handling new appointments that are pending confirmation

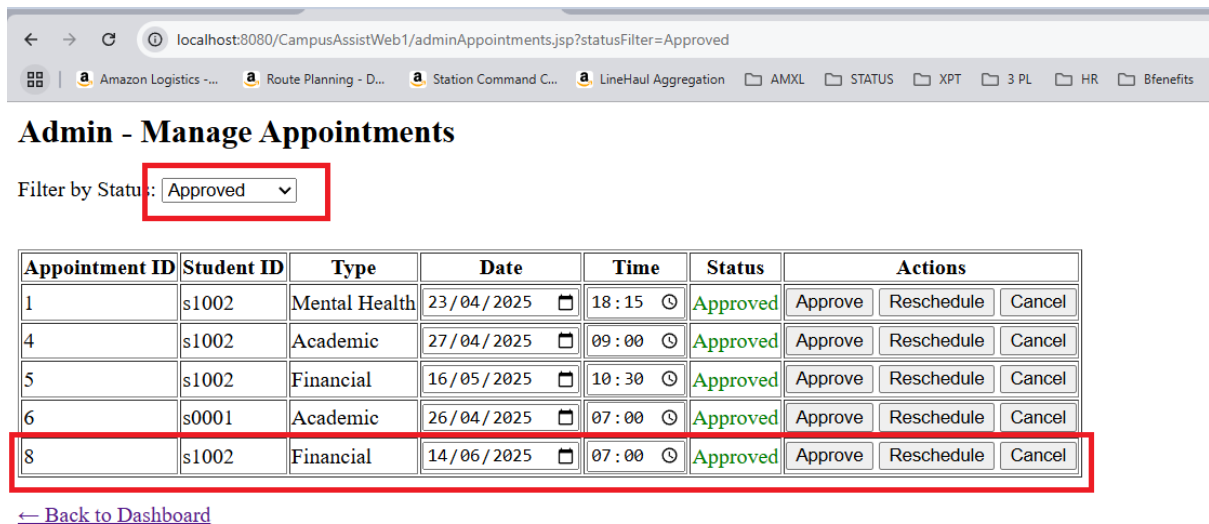


Figure 33 - Appointment After Admin Approval

Figure 32 shows the identical appointment once the admin clicks "Approve." The current status is now shown as Approved and is highlighted in green. The filter has also been adjusted to "Approved" to separate verified appointments for more straightforward tracking, but the admin can always switch back to deal with other appointments.

3.4. The new code developed for AE2

The AE2 stage of the CampusAssist initiative necessitated shifting from a console application to a comprehensive web platform utilizing JSP, Servlets, JDBC, and SQLite. Consequently, all code associated with the web interface was newly created for AE2, encompassing session management, dynamic form generation, and database interactions through prepared statements.

Key newly developed components include:

- **Student Functionality (Member A):**
 - 🔗 bookAppointment.jsp, giveFeedback.jsp, studentDashboard.jsp, studentViewFeedback.jsp
 - 🔗 SubmitFeedbackServlet.java - manages secure feedback integration with database connections
 - 🔗 Logic for student login based on sessions to customize the dashboard and restrict unauthorized entry
- **Admin Functionality (Member B):**
 - 🔗 adminDashboard.jsp - core center for administrative functions;
 - 🔗 adminAppointments.jsp - interactive table with color-highlighted statuses and rescheduling capabilities;

- ✚ AdminResponseServlet.java - permits administrators to reply to and classify feedback;
- ✚ adminFaq.jsp – CRUD interface for FAQ management;
- ✚ adminAnalytics.jsp – generates visual summaries and charts using HTML/CSS.
- **FAQ and Analytics (Member C):**
 - ✚ FaqServlet.java – manages insertion, updating, and deletion of FAQs;
 - ✚ feedbackChart.jsp, appointmentStats.jsp – visualize key feedback metrics for decision making;

Along with the user interface and controller servlets, new database tables were developed or enhanced:

- ✚ feedback table now includes response and category fields;
- ✚ appointments table includes a status field to enable admin workflows;

Alongside the user interface and controller servlets, new database tables were either created or modified:

Across the AE2 codebase, secure JDBC techniques like prepared statements, session verification, and effective error management were employed to guarantee both stability and security.

Screenshots and code examples are presented in this report (refer to Figures 12–34) to demonstrate the layout and execution of the new components.

4. Principles of Software Engineering

4.1. Design modularity and separation of concerns

The project utilized a modular design, distributing tasks among various Java classes, JSPs, and Servlets. Every module manages a distinct function:

- JSP files manage the user interface and presentation layer;
- Servlets (like SubmitFeedbackServlet.java, AdminResponseServlet.java) manage business operations;
- JDBC handles all database interactions, separated from the presentation layer through prepared statements;

```

SubmitFeedbackServlet.java
Source History
1 package servlets;
2
3 import java.io.*;
4 import java.sql.*;
5 import jakarta.servlet.*;
6 import jakarta.servlet.http.*;
7
8 public class SubmitFeedbackServlet extends HttpServlet {
9     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
10         HttpSession session = request.getSession(false);
11         String studentId = (session != null) ? (String) session.getAttribute("studentId") : null;
12
13         String appointmentId = request.getParameter("appointmentId");
14         String feedbackText = request.getParameter("feedbackText");
15
16         if (studentId == null || appointmentId == null || feedbackText == null) {
17             response.getWriter().println("Missing required information.");
18             return;
19         }
20
21         try {
22             Class.forName("org.sqlite.JDBC");
23             Connection conn = DriverManager.getConnection("jdbc:sqlite:C:/Users/pirico/Desktop/Uni/OODD projects/OODD group/CampusAssist/campusassist.db");
24
25             String sql = "INSERT INTO feedback (appointmentID, studentID, feedbackText) VALUES (?, ?, ?)";
26             PreparedStatement stmt = conn.prepareStatement(sql);
27             stmt.setInt(1, Integer.parseInt(appointmentId));
28             stmt.setString(2, studentId);
29             stmt.setString(3, feedbackText);
30             stmt.executeUpdate();
31
32             conn.close();
33             response.sendRedirect("studentDashboard.jsp?feedbackStatus=success");
34         } catch (Exception e) {
35             e.printStackTrace();
36             response.getWriter().println("Error submitting feedback: " + e.getMessage());
37         }
38     }
39 }

```

Figure 34 - Feedback Submission Servlet (SubmitFeedbackServlet.java)

This illustration shows the modular backend system designed for managing feedback submissions. The servlet gathers session and form information, checks input validity, and employs JDBC to add feedback to the database. This encapsulation guarantees that business logic is distinct from the user interface.

```

SubmitFeedbackServlet.java giveFeedback.jsp
Source History
1 <%% page import="java.sql.*, jakarta.servlet.*" %>
2 <%% page contentType="text/html; charset=UTF-8" language="java" %>
3
4 String studentId = (session != null) ? (String) session.getAttribute("studentId") : null;
5
6 if (studentId == null) {
7     response.sendRedirect("login.jsp");
8     return;
9 }
10
11 // Connect to DB and get student's appointments
12 Connection conn = null;
13 PreparedStatement stmt = null;
14 ResultSet rs = null;
15
16 <html>
17 <head><title>Give Feedback</title></head>
18 <body>
19 <h2>Give Feedback for Your Appointments</h2>
20
21 <form method="post" action="SubmitFeedbackServlet">
22     <label for="appointmentId">Select Appointment:</label>
23     <select name="appointmentId" required>
24
25         <%%
26         try {
27             Class.forName("org.sqlite.JDBC");
28             conn = DriverManager.getConnection("jdbc:sqlite:C:/Users/pirico/Desktop/Uni/OODD projects/OODD group/CampusAssist/campusassist.db");
29             String sql = "SELECT id, type, date, time FROM appointments WHERE studentID=?";
30             stmt = conn.prepareStatement(sql);
31             stmt.setString(1, studentId);
32             rs = stmt.executeQuery();
33
34             while (rs.next()) {
35
36                 <option value="<%= rs.getInt("id") %>">
37                     <%= rs.getString("type") %> on <%= rs.getString("date") %> at <%= rs.getString("time") %>
38                 </option>
39             }
40         } catch (Exception e) {
41             out.println("Error loading appointments: " + e.getMessage());
42         } finally {
43             if (rs != null) try { rs.close(); } catch (Exception ignored) {}
44             if (stmt != null) try { stmt.close(); } catch (Exception ignored) {}
45         }
46     </select>
47     <input type="text" name="feedbackText" />
48     <input type="submit" value="Submit Feedback" />
49 </form>
50
51 <%%
52 try {
53     Class.forName("org.sqlite.JDBC");
54     conn = DriverManager.getConnection("jdbc:sqlite:C:/Users/pirico/Desktop/Uni/OODD projects/OODD group/CampusAssist/campusassist.db");
55     String sql = "INSERT INTO feedback (appointmentID, studentID, feedbackText) VALUES (?, ?, ?)";
56     PreparedStatement stmt = conn.prepareStatement(sql);
57     stmt.setInt(1, Integer.parseInt(appointmentId));
58     stmt.setString(2, studentId);
59     stmt.setString(3, feedbackText);
60     stmt.executeUpdate();
61     conn.close();
62     response.sendRedirect("studentDashboard.jsp?feedbackStatus=success");
63 } catch (Exception e) {
64     e.printStackTrace();
65     response.getWriter().println("Error submitting feedback: " + e.getMessage());
66 }
67
68 </body>
69 </html>

```

Figure 35 - Feedback Form JSP Page (giveFeedback.jsp)

This JSP file displays the feedback interface for the student. It retrieves the student's appointment information from the database and sends the form to the SubmitFeedbackServlet. This distinct division between frontend and backend corresponds with the separation of concerns principle in software development.

4.2. MVC Implementation

The CampusAssist system adheres to the Model-View-Controller pattern:

- **Model:** SQLite database schema (appointments, feedback, students);
- **View:** JSP pages such as studentDashboard.jsp, adminFaq.jsp, etc.
- **Controller:** Servlets such as AppointmentActionServlet, FaqServlet;

This pattern was consistently followed across all aspects to guarantee maintainability and scalability.

```

1  <%@ page import="java.sql.*, jakarta.servlet.*" %>
2  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
3
4  <%
5      String studentId = (String) session.getAttribute("studentId");
6      String role = (String) session.getAttribute("role");
7
8      if (studentId == null || !"student".equals(role)) {
9          response.sendRedirect("login.jsp");
10         return;
11     }
12
13     Connection conn = null;
14     PreparedStatement stmt = null;
15     ResultSet rs = null;
16
17 <!-- DOCTYPE html -->
18 <html lang="en">
19 <head>
20     <meta charset="UTF-8">
21     <title>Student Dashboard - CampusAssist</title>
22     <style>
23         body {
24             font-family: "Segoe UI", Tahoma, sans-serif;
25             background-color: #f0f6fb;
26             padding: 40px;
27             text-align: center;
28         }
29
30         h2 {
31             color: #003366;
32         }
33
34         h3 {
35             color: #004d99;
36             margin-top: 30px;
37         }
38
39         table {
40             margin: 0 auto;
41             border-collapse: collapse;
42             width: 80%;
43             background-color: #fff;
44             box-shadow: 0 0 8px rgba(0,0,0,0.1);
45             margin-top: 20px;

```

Figure 36 - View Component (studentDashboard.jsp)

This illustration displays the JSP page that is in charge of generating the student dashboard. It functions as the View in the MVC framework, presenting appointment information sourced from the model layer and allowing user engagement through action buttons. The code

segment features session verification and integrated HTML/CSS design for arrangement and functionality.

```

1 package servlets;
2
3 import java.io.*;
4 import java.sql.*;
5 import jakarta.servlet.*;
6 import jakarta.servlet.http.*;
7
8 public class AppointmentActionServlet extends HttpServlet {
9     protected void doPost(HttpServletRequest request, HttpServletResponse response)
10         throws ServletException, IOException {
11
12         int appointmentId = Integer.parseInt(request.getParameter("appointmentId"));
13         String action = request.getParameter("action");
14         String date = request.getParameter("date");
15         String time = request.getParameter("time");
16
17         String newStatus = "";
18         if ("Approve".equals(action)) newStatus = "Approved";
19         else if ("Reschedule".equals(action)) newStatus = "Rescheduled";
20         else if ("Cancel".equals(action)) newStatus = "Cancelled";
21
22         try {
23             Class.forName("org.sqlite.JDBC");
24             Connection conn = DriverManager.getConnection("jdbc:sqlite:C:/Users/piricoand/Desktop/Uni/OODD projects/OODD group/CampusAssist/campusassist.db");
25
26             String sql = "UPDATE appointments SET date = ?, time = ?, status = ? WHERE id = ?";
27             PreparedStatement stmt = conn.prepareStatement(sql);
28             stmt.setString(1, date);
29             stmt.setString(2, time);
30             stmt.setString(3, newStatus);
31             stmt.setInt(4, appointmentId);
32
33             stmt.executeUpdate();
34             stmt.close();
35             conn.close();
36
37             response.sendRedirect("adminAppointments.jsp?statusFilter=" + newStatus); // Optional
38         } catch (Exception e) {
39             throw new ServletException("Database update failed", e);
40         }
41     }
42 }

```

Figure 37 - Controller Component (AppointmentActionServlet.java)

This Java servlet processes requests to approve, reschedule, or cancel appointments, serving as the Controller within the MVC framework. It handles user input from the interface, uses logic to establish the new status, and modifies the model (database) appropriately. The servlet facilitates modular routing and distinct separation of responsibilities.

| id | studentID | type | date | time | details | status |
|----|-----------|---------------|------------|-------|---|-------------|
| 1 | s1002 | Mental Health | 2025-04-23 | 18:15 | Insomnia. | Approved |
| 2 | s1002 | Mental Health | 2025-04-25 | 03:14 | Anxiety. | Cancelled |
| 3 | s1002 | Mental Health | 2025-04-23 | 06:15 | Insomnia. | Rescheduled |
| 4 | s1002 | Academic | 2025-04-27 | 09:00 | Hello,... | Approved |
| 5 | s1002 | Financial | 2025-05-16 | 10:30 | Hello,... | Approved |
| 6 | s0001 | Academic | 2025-04-26 | 07:00 | Hello, please let me know when you have a financial advisor available. Thank you! | Approved |

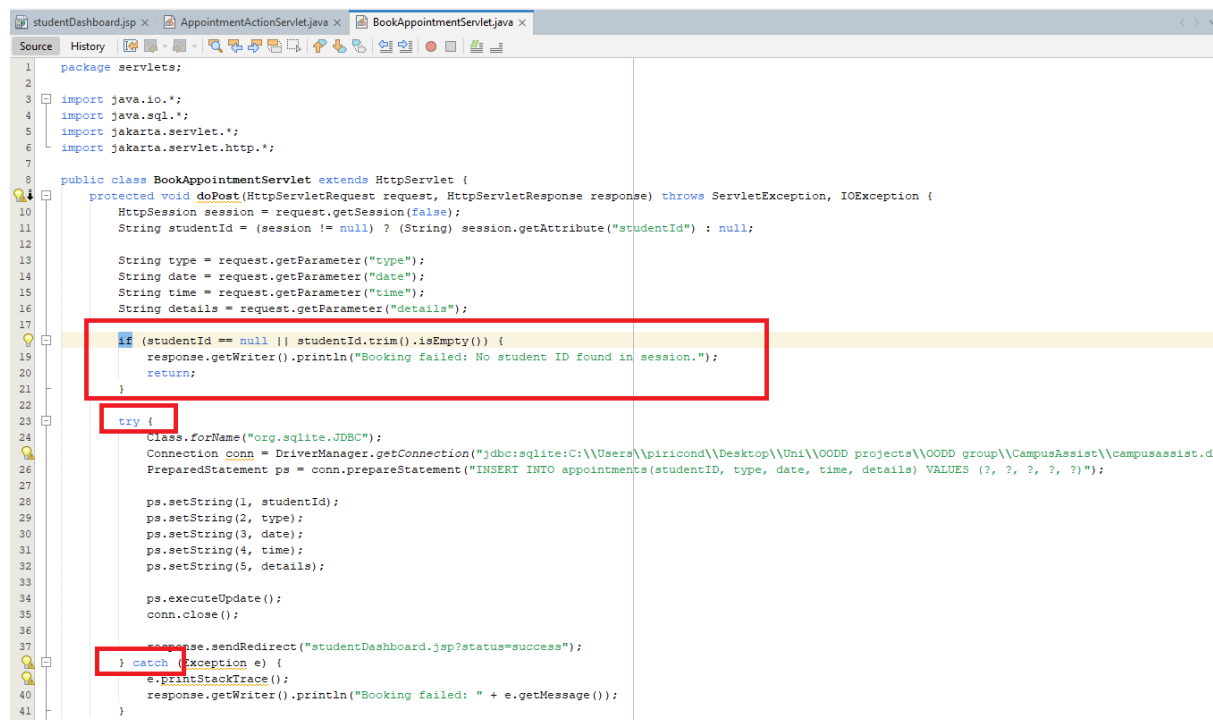
Figure 38 - Model Layer (appointments table in SQLite)

This illustration displays the appointments table from the SQLite database, embodying the Model in the MVC architecture. It retains ongoing information including student ID, type of appointment, date, time, specifics, and current status. Controller servlets such as AppointmentActionServlet.java access and manipulate this data.

4.3. Validation and error handling

Error management was executed via:

- Enclose all JDBC operations within try-catch blocks;
- Verification of session attributes to avoid unauthorized access;
- Input validation for forms through the use of required attributes and logical checks in servlets.



This illustration showcases the application of input validation and organized error management in the BookAppointmentServlet. Prior to adding appointment information to the database, the servlet verifies that the studentId exists and is not empty. If validation is unsuccessful, the request stops, and an appropriate error message is provided. The try-catch structure manages SQL exceptions smoothly, displaying error information and preventing the application from crashing abruptly. This strengthens durability and enhances system dependability by blocking invalid actions and transparently recording errors.

5. Evaluation and testing

5.1. Test plan JUnit

Because the CampusAssist system is web-based, essential business logic was integrated into JSP and Servlet files, complicating unit testing with JUnit. Nonetheless, for backend classes like AppointmentManager.java and FeedbackManager.java (if they are distinct), JUnit tests may be created to validate:

- Adding and accessing appointment information;
- Logic for filtering the status of appointments;
- Feedback text classification (if modularized);

Note! Due to strong servlet interconnection and direct database access, the majority of testing was performed via integrated system and browser-based functional validation.

5.2. Functionality testing with Tomcat

All features visible to users were extensively verified by launching the application in Apache Tomcat 10.1.39 and reaching it through localhost:8080. This includes:

- Signing in as a student/admin;
- Scheduling appointments;
- Submitting and reviewing feedback;
- Management of appointments, feedback, FAQs, and analytics by the admin;
- Refining and revising statuses.

The project was implemented and evaluated locally with Apache Tomcat 10.1.39. Functional tests were conducted by mimicking usual user pathways for both students and administrators. A key feature evaluated is appointment management, which encompasses student-side booking and status oversight from the admin panel.

Support Type: Mental Health

Date: 26/06/2025

Time: 07:30

Details:

Please book a therapy session as part of the student free mental health program. Thank you!

Book Appointment

Figure 40 - Student Booking an Appointment (Mental Health)

This illustration displays the bookAppointment.jsp interface designed for students. Student s1001 chooses "Mental Health" as the type of support, fills in a date, time, and provides extra notes asking for a therapy session. This form links to the BookAppointmentServlet.java, which adds the data to the appointments table when submitted.

For instance I can introduce from student's s1001 perspective a mental health booking.

| Welcome, Student s1001! | | | |
|-------------------------|------------|-------|---|
| Your Appointments: | | | |
| Type | Date | Time | Details |
| Mental Health | 2025-06-26 | 07:30 | Please book a therapy session as part of the student free mental health program. Thank you! |

Figure 41 - Dashboard Displaying Student's Booking

Here, on the student's dashboard, we can see that the request has been succesful. Once submitted, the dashboard updates and displays the newly added appointment in the table. This verifies that the booking feature is functional and the appointment has been correctly saved in the database. The real-time update showcases the connection between frontend JSP and backend servlet functionality.

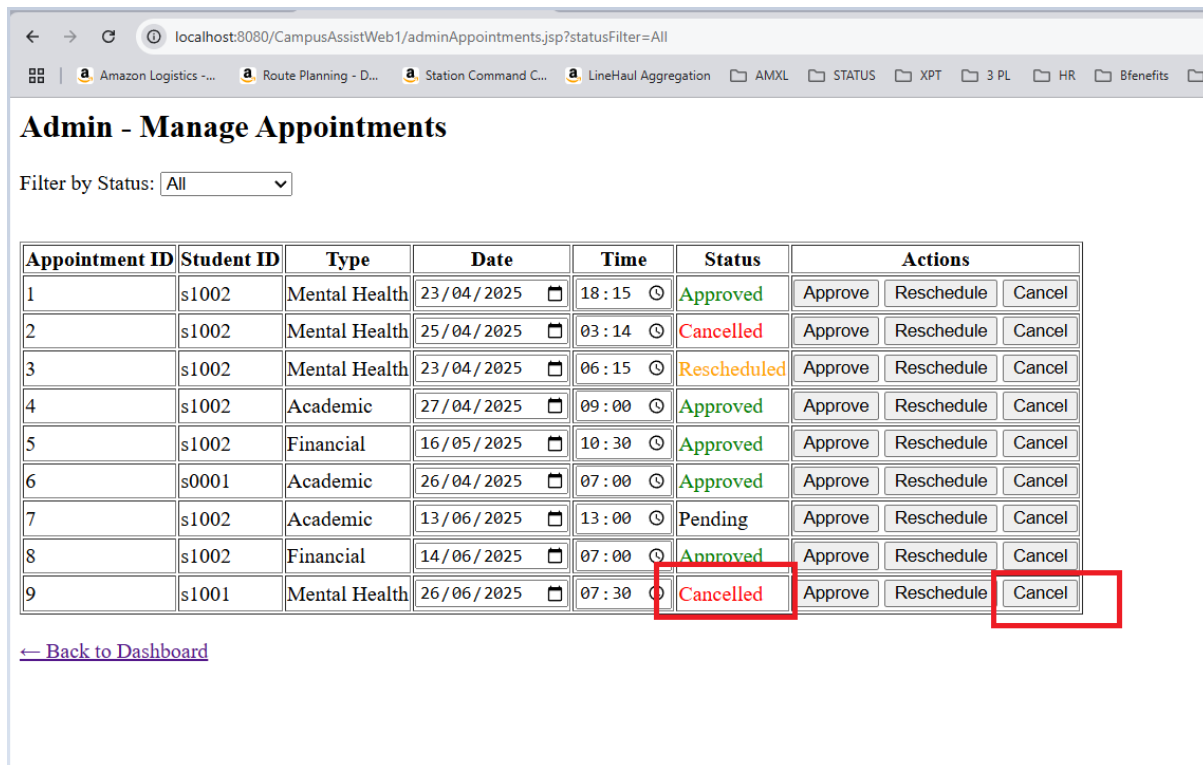


Figure 42 - Admin Cancelling the Appointment (Test Scenario)

To check the functionality from the admin side, the fresh booking is reviewed via `adminAppointments.jsp`. The administrator can utilize the form buttons to filter, approve, reschedule, or cancel the entry. In this instance, the administrator has terminated the appointment to showcase the feature. The status column changes to `Cancelled`, and the entry stays visible for tracking purposes.

5.3. Verification of Functions and Corrections

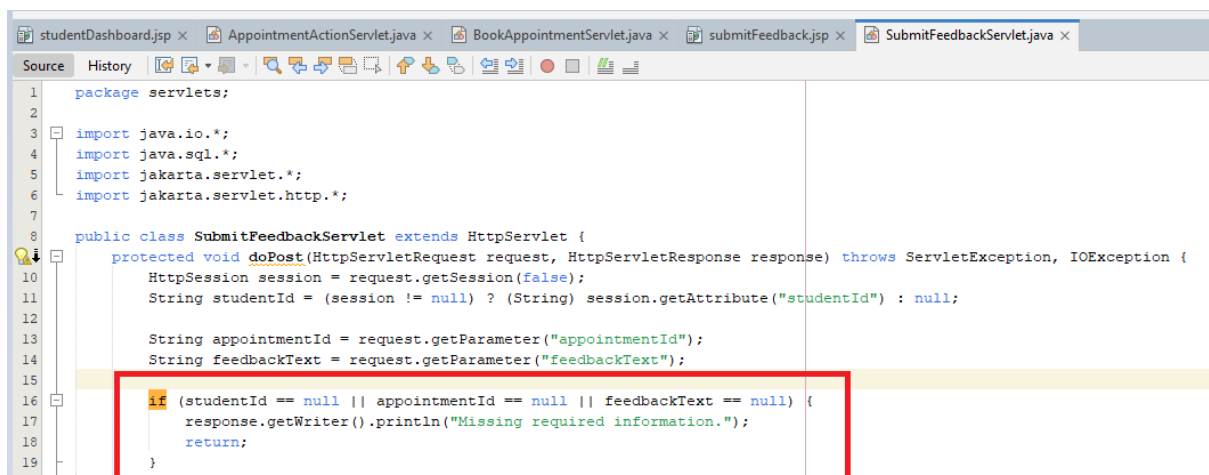
To confirm that functionality was implemented properly, every AE2 feature underwent validation via integration and UI tests in the local Tomcat setup. The verifications affirmed that:

- ✓ Students are able to effectively schedule appointments using the `bookAppointment.jsp` form.
- ✓ Feedback submission links accurately to a chosen appointment and is stored in the database through `SubmitFeedbackServlet.java`.
- ✓ Admins have the ability to view all appointments in `adminAppointments.jsp` and take actions: Approve, Reschedule, or Cancel.

- ✓ Changes in status are instantly reflected in both the database and the UI (the status column updates with color coding)
- ✓ The admin feedback response feature is operational, with the ability to save responses and categories through AdminResponseServlet.java.
- ✓ All pages adhere to session-based access (students are unable to access admin pages and vice versa).
- ✓ The error-handling logic was tested by eliminating parameters or disrupting database paths to assess fallback messaging

Correct Validation Illustrations:

Fix 1: Initially, submitting feedback without choosing an appointment led to null pointer errors. This issue was resolved by verifying all input fields prior to processing (SubmitFeedbackServlet.java);



```

1 package servlets;
2
3 import java.io.*;
4 import java.sql.*;
5 import jakarta.servlet.*;
6 import jakarta.servlet.http.*;
7
8 public class SubmitFeedbackServlet extends HttpServlet {
9     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
10         HttpSession session = request.getSession(false);
11         String studentId = (session != null) ? (String) session.getAttribute("studentId") : null;
12
13         String appointmentId = request.getParameter("appointmentId");
14         String feedbackText = request.getParameter("feedbackText");
15
16         if (studentId == null || appointmentId == null || feedbackText == null) {
17             response.getWriter().println("Missing required information.");
18             return;
19         }
20     }
21 }

```

Figure 43 - Submitfeedback fix

Fix 2: Student dashboards used to show all feedback without being filtered for individual students. The SQL query was modified to include a WHERE studentID=? clause to display only pertinent entries

Fix 3: Status colors (green, orange, red) were absent. A new utility function getColorForStatus() was created and visually tested in adminAppointments.jsp

Admin - Manage Appointments

Filter by Status: All

| Appointment ID | Student ID | Type | Date | Time | Status | Actions |
|----------------|------------|---------------|----------------|---------|-------------|---------------------------|
| 1 | s1002 | Mental Health | 23 / 04 / 2025 | 18 : 15 | Approved | Approve Reschedule Cancel |
| 2 | s1002 | Mental Health | 25 / 04 / 2025 | 03 : 14 | Cancelled | Approve Reschedule Cancel |
| 3 | s1002 | Mental Health | 23 / 04 / 2025 | 06 : 15 | Rescheduled | Approve Reschedule Cancel |
| 4 | s1002 | Academic | 27 / 04 / 2025 | 09 : 00 | Approved | Approve Reschedule Cancel |
| 5 | s1002 | Financial | 16 / 05 / 2025 | 10 : 30 | Approved | Approve Reschedule Cancel |
| 6 | s0001 | Academic | 26 / 04 / 2025 | 07 : 00 | Approved | Approve Reschedule Cancel |
| 7 | s1002 | Academic | 13 / 06 / 2025 | 13 : 00 | Pending | Approve Reschedule Cancel |
| 8 | s1002 | Financial | 14 / 06 / 2025 | 07 : 00 | Approved | Approve Reschedule Cancel |
| 9 | s1001 | Mental Health | 26 / 06 / 2025 | 07 : 30 | Cancelled | Approve Reschedule Cancel |

[← Back to Dashboard](#)

Figure 44 - Status Colors

These evaluations verified that every requirement from the AE2 specification was properly implemented and is operating as expected within the integrated web environment.

6. Conclusions and possible future improvements

6.1. Summary of achievements

The AE2 phase of the CampusAssist initiative effectively converted the original group-oriented Java console application into a complete web platform employing JSP, Servlets, JDBC, and SQLite. The application currently accommodates both student and administrator roles, offering customized features and secure access through session management.

Notable accomplishments comprise:

- Web Conversion: All AE1 features—appointment scheduling, feedback collection, FAQs—were transferred to the web through JSP and Servlets.
- Session Management: Login systems for students and administrators were established with session-oriented redirection and access limitations based on roles.
- Student Attributes: Students are able to schedule support meetings, provide feedback, check appointment history, and read support FAQs

- Admin Functions: Administrators have the ability to approve, cancel, or reschedule appointments, manage and respond to feedback, and interpret data using dashboards and charts.
- UI and UX: CSS was applied to pages for improved clarity and usability; dynamic components show real-time updates, like appointment status or submitted feedback.
- Data Persistence: A persistent SQLite database is used to store and query all data through prepared JDBC statements for stability and security purposes.
- Modularity and MVC: The project was organized based on modular principles, division of tasks, and MVC architecture, improving maintainability.
- Validation and Testing: Functional testing within Apache Tomcat confirmed system integrity. Corrections were made according to feedback on error management and user experience

6.2. Possible improvements

Although AE2 has been successful, there are various enhancements that could be implemented to advance CampusAssist to a production-quality platform:

- Password Encryption: At present, passwords are kept in plain text. Integrating hashing algorithms (such as BCrypt) would significantly improve security.
- Pagination and Sorting: To enhance usability, appointment and feedback tables with pagination would enable quicker browsing and improved data management.
- Email Alerts: Students might get automatic email confirmations for reservations, authorizations, or terminations
- Dynamic Analytics: Rather than relying on fixed tables, admin analytics could leverage interactive charts and filters through JavaScript libraries such as Chart.js.
- Sentiment Analysis of Feedback: A unified NLP-driven tool could assess written feedback for tone or emotional classification, surpassing manual labeling.
- Mobile Responsiveness: An adaptable design would enhance accessibility for students using smartphones or tablets.
- Admin Dashboard Roles: Upcoming versions may feature hierarchical roles for admins (e.g., viewer-only, feedback-manager-only)

7. References

Oracle (2024). *Java Platform, Standard Edition Documentation*.

<https://docs.oracle.com/javase/>

Apache Software Foundation (2024). *Apache Tomcat 10.1 Documentation*.

<https://tomcat.apache.org/tomcat-10.1-doc/>

SQLite Consortium (2024). *SQLite Official Documentation*.

<https://www.sqlite.org/docs.html>

W3Schools (2024). *JSP and Servlet Tutorials*. <https://www.w3schools.com/jsp/>

GeeksforGeeks (2024). *Java JDBC Examples*. <https://www.geeksforgeeks.org/jdbc-in-java/>

Draw.io (2024). *UML Diagrams and Software Modelling Tool*. <https://app.diagrams.net/>

Solent University (2024). *QHO543 Object Oriented Design and Development AE2 Brief*.