SC2002 - Object-Oriented Design & Programming

Assignment : Internship Placement Management System

Lab group	SCEA
Members	Junchan Koompa (U2420662E) Tongdon-ngao Plengpin (U2420420D) Satsatranurak Phattarachai (U2420262F) Naresh Pradhyun (U2422594A) Sundhar Triya Harshini (U2423395A)

1. Design Considerations

1.1 Design Approach and Considerations

The Internship Placement Management System follows a layered MVC architecture, separating domain entities, business logic, and UI. This structure cleanly manages interactions among students, company representatives, and career center staff.

- Model (Entities): An abstract User class is extended by Student, CompanyRepresentative, and CareerCenterStaff. Core entities (Internship, InternshipApplication) enforce their own lifecycle rules using enums such as ApplicationStatus, InternshipLevel, and InternshipStatus, ensuring type safety and valid state transitions.
- Controllers (Business Logic): All validation and rules are implemented independently of the UI. AuthController: authentication and password changes. StudentController: eligibility, limits, and placement. CompanyController: internship validation and approvals. StaffController: system-wide approvals and reporting. Controllers return standardized Result objects to keep success/error handling uniform and UI-agnostic.
- Boundaries (Presentation Layer): The system offers both a console UI and a modular Swing GUI. Each user role has a dashboard composed of a main frame, filterable table panels, and an ActionHandler that routes events to controllers. A shared login frame authenticates users and redirects them accordingly.

The MVC split enables new interfaces to reuse existing logic. The abstract User class and enums support adding new roles and states. The GUI was decomposed from a monolith into focused components for clarity and testability. SRP is applied across controllers, panels, and handlers. Some validation exists in both UI and controllers for responsiveness, but controllers remain the source of truth. The ActionHandler adds minor overhead but improves modularity.

Key patterns include Result Object, Action Handler/Command, Table Model, and Enum-based state machines; heavier patterns (Repository, Observer, Builder) were intentionally avoided to preserve simplicity. Overall, the architecture remains clear, modular, and extensible.

1.2 Principles Used

1.2.1 Basic Principles : The system applies core OO principles throughout. Encapsulation is enforced with private fields and controlled accessors. Abstraction is provided by the abstract User class, which centralizes shared authentication while subclasses define role-specific behavior. Inheritance establishes clear IS-A relationships among the three user types, and polymorphism enables uniform authentication and automatic routing to the correct dashboards. Composition models real-world links—InternshipApplication contains a Student and an Internship, and each Internship contains its CompanyRepresentative owner. Together, these principles produce a clear and natural object model.

1.2.2 SOLID Principle application : Our project follows SOLID design principles to keep the system modular, maintainable, and easy to extend.

Example implementations for each principle are shown below :

- **Single Responsibility Principle :** The system follows SRP by ensuring each class has one clear purpose. In the controller layer, StudentController manages student-specific logic CompanyController handles internship posting and application approval, and StaffController focuses solely on administrative approvals.
- **Open/ Close Principle :**The system follows the Open/Closed Principle (OCP) by allowing extension without modifying existing code. The abstract User class exemplifies this: new user roles can be added by simply creating subclasses such as Admin extends User without altering the original class.
- **Liskov Substitution Principle :** The system follows the Liskov Substitution Principle (LSP) through its User hierarchy. Student, CompanyRepresentative, and CareerCenterStaff can all be used wherever a User is expected without breaking functionality.
- **Interface Segregation Principle :** The project applies ISP through interfaces like IUserRepository and IInternshipFilter. Normal classes will not have to do the database loading task. Controllers that don't need filtering don't depend on it. One implementation serves all controllers.

- **Dependency Injection Principle :** The system applies the Dependency Inversion Principle (DIP) by ensuring high-level modules depend on abstractions rather than concrete classes. For example, in MainApp, polymorphic routing relies on the abstract User type rather than specific subclasses; currentUser is declared as User, not Student or Staff. Concrete subclasses also depend on the same abstraction, reducing coupling and allowing new user roles to be added without modifying the main logic.

1.3 Assumptions Made

1.3.1 Single-User Operation: The system runs in single-user mode with no concurrency, so no thread-safety is needed. Data is loaded once at startup and kept in memory, simplifying the design while demonstrating core OOP ideas.

1.3.2 Static Sample Data: Initial data comes from pre-validated CSV files and is not saved after the application ends, allowing the project to focus on object-oriented structure and business rules instead of database integration.

2. UML Class Diagram

Description: We identified main classes by analyzing system requirements: User, Student, CompanyRepresentative, CareerCenterStaff, Internship, InternshipApplication, and four enums. The **Entity package** contains domain objects with business rules (max 3 applications per student, max 5 internships per representative). The **Controller** package centralizes business logic through static utility classes (AuthController, StudentController, CompanyController, StaffController). The **Boundary** package provides dual interfaces—console menus and GUI dashboards using Java Swing. The Main package contains entry points (MainApp, MainAppGUI) and DataManager for CSV loading. We used inheritance (User hierarchy), composition (Dashboard owns Panels), and aggregation (Panels share ActionHandler). This layered architecture ensures modularity, maintainability, and clear separation of concerns.

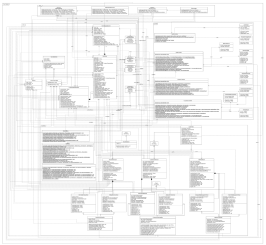


Figure 1 Class Diagram (refer to github for clearer image)

3. UML Sequence Diagram

Description: We analyzed **four** key workflows from use cases.

1. **Dashboard Access & Retrieve Pending Applications:** shows CompanyRepresentative accessing CompanyMenu, which calls CompanyController to filter pending applications by status, looping through internships to retrieve associated applications.
2. **Company Representative Checks Updated Internship Status:** demonstrates refreshing internship lists, calling CompanyController to get updated internships with available slots and FILLED status.
3. **Student Accepts Placement:** illustrates StudentMenu calling PlacementController to set application status to ACCEPTED, confirm placement, update internship slots, check if filled, and display confirmation.
4. **Process Application (Approve/Reject):** shows CompanyMenu retrieving applications, validating selections, then calling ApplicationController to set status (SUCCESSFUL/UNSUCCESSFUL) and display results. These workflows demonstrate clear separation between boundary, controller, and entity layers with proper validation and state management.

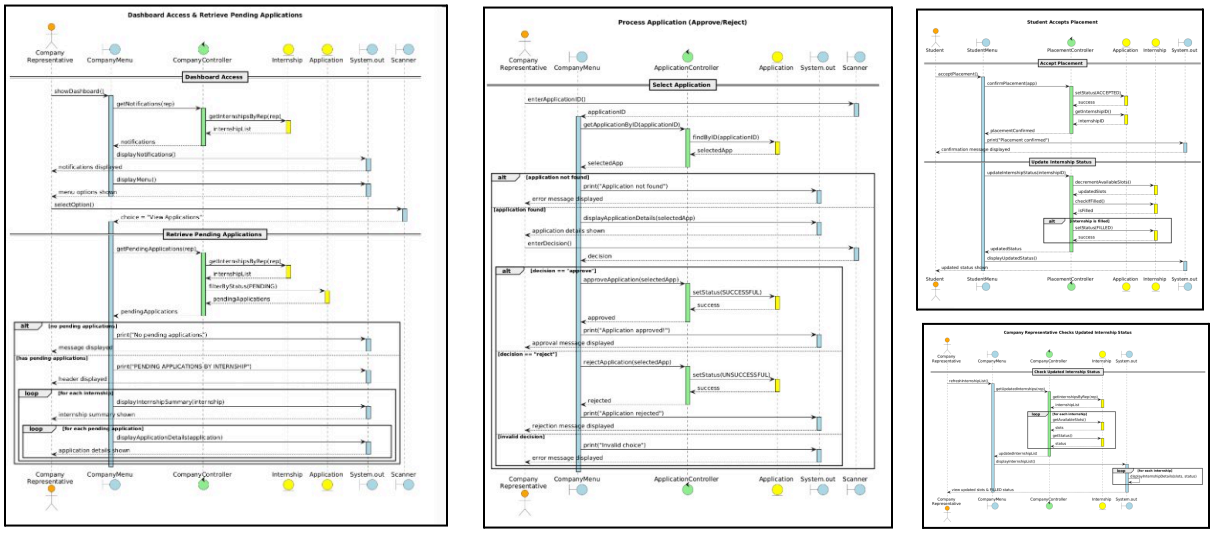


Figure 2 Sequence Diagrams (refer to github for clearer image)

4. Additional Features/Functionalities

4.1 GUI Implementation

The project has full swing-based GUI with:

- Login frame with green theme
- Tabbed dashboards for each role
- Split panels (Student, Company, Staff)
- Tables with custom rendering (color-coded rows)
- Rounded buttons (RoundedButton.java)
- Action handlers for dialogs and validation

4.2 Real-Time Dashboard Refresh

Tables auto-refresh after actions (approve/reject/withdraw/delete). Filters re-apply after refresh. The table also has a refresh button to manually refresh the data. This helps improve user experience; users don't need to logout/login to see changes. Example:

Figure 3 Refresh Button

4.3 Tooltip Help Text

Buttons have tooltips (e.g., “Apply to Internship” → “Select an internship from the table and click to apply”) and fields have tooltips (e.g., “Format: YYYY-MM-DD (leave blank for all)”) Examples:

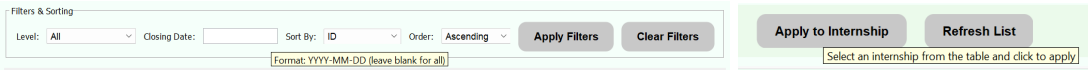


Figure 4 Tooltip Help Texts

4.4 Report Generation Panel

StaffDashboard has Reports tab with buttons:

- Applications Report
- Companies Report
- Full System Report
- Pending Internships
- Pending Withdrawals
- Approved Internships

Each button shows a scrollable text area with formatted reports. This helps staff quickly audit system state.

4.5 Dialog Box

The system displays a dialog box after users click a button, to confirm action or to notify successful action. Examples:

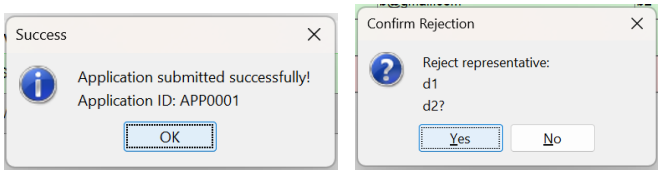


Figure 5 Dialog Box (refer to github for full images)

4.6 Filters and Sorting

The system provides filters and sorting functions for internships. Users can filter by status, level, and/or company, and can sort by Default (Status), ID, title, company, level, opening date, and closing date, either ascending or descending.

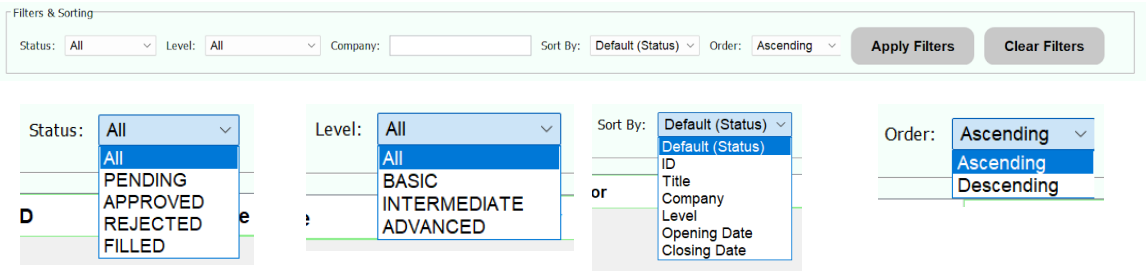
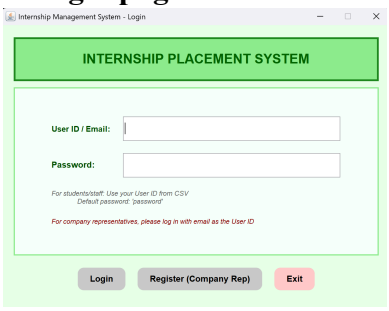


Figure 6 Filters and Sorting function

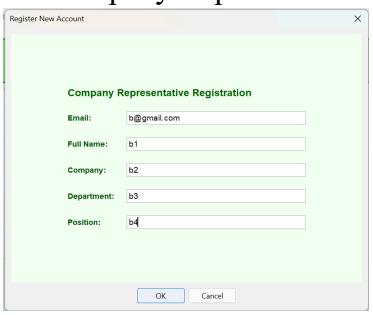
5. Results

5.1 User Interfaces

1. Login page



1.1 Company Representative Registration

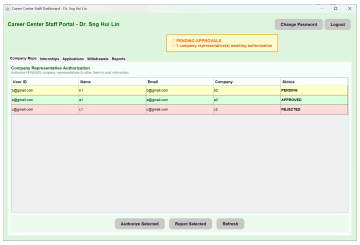


The system provides an interface for all users to **log in**. Authentication is performed using a Student ID, NTU account (Staff), or company email (Company Representative).

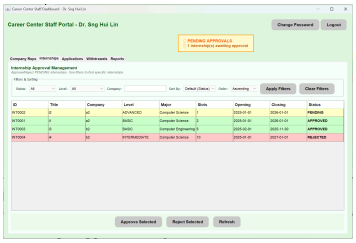
- **Company Representative Registration:** The system allows a new Company Representative to submit a registration request. This creates an account with a pending status which requires authorization from a Career Center Staff member.

2. Career Center Staff Dashboard

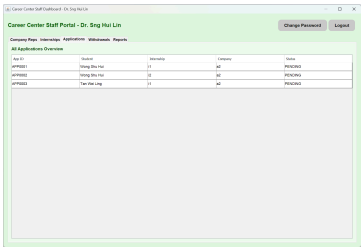
2.1 Company Representative Authorization



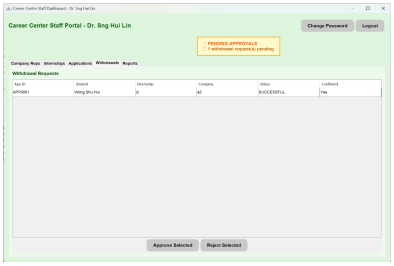
2.2 Internship Approval Management



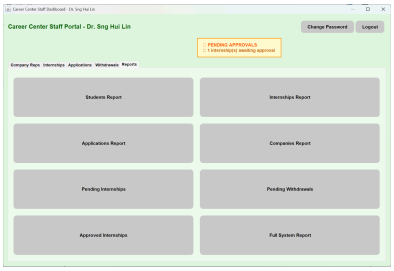
2.3 Applications



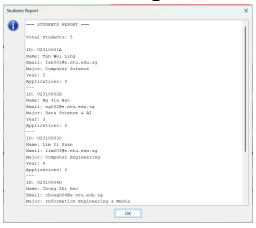
2.4 Withdrawals



2.5 Reports



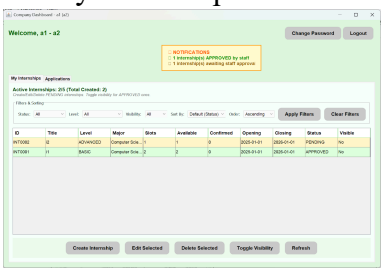
2.5.1 Example of Reports: Students Reports



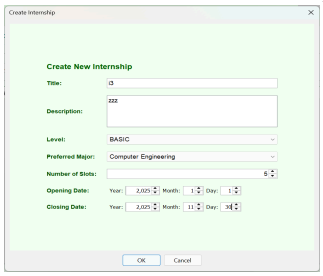
- **Authorize Company Representatives:** View and filter all representative accounts. Staff can approve or reject pending accounts, and the decision is final.
- **Approve Internship Opportunities:** View internships with filters (status, level, major). Staff can approve or reject any internship marked **PENDING**.
- **View Applications:** Display all student-submitted applications.
- **Process Withdrawal Requests:** Show all student withdrawal requests. Staff approve or reject each one; approved withdrawals update the application and free the internship slot if already confirmed.
- **Generate Reports:** Staff can generate filtered reports on students, internships, applications, pending items, and more.

3. Company Dashboard

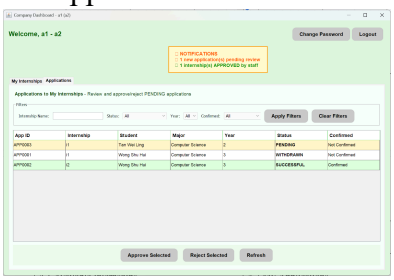
3.1 My Internships



3.1.1 Select Create Internship



3.2 Applications



Manage Internship Listings:

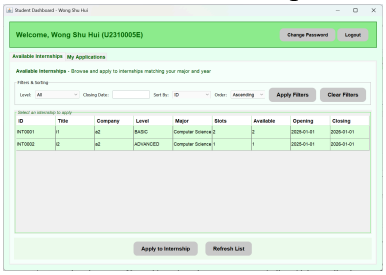
- **Create:** Representatives can create new internship listings, up to a maximum of 5 active opportunities (Pending/Approved, not Filled or expired).
- **Edit/Delete:** The system only permits the editing or deleting of internships with a "PENDING" status.
- **Toggle Visibility:** For internships with an "APPROVED" status, the representative can toggle the visibility on or off, controlling whether the listing is visible to students.

Process Student Applications:

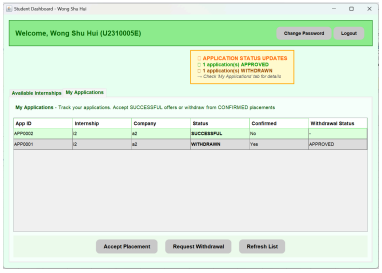
- **View and Filter:** The system displays all applications with filters for internship, status, student year, and confirmation status.
- **Approve/Reject:** The representative can review student details and either approve ("SUCCESSFUL") or reject ("UNSUCCESSFUL") their applications.
- **Slot Constraint:** The system prevents a representative from approving more applications than the number of available slots (displayed on the "My Internships" tab). Available slots are calculated as Total Slots - Number of 'SUCCESSFUL' Applications.

4. Student Dashboard

4.1 Available Internships



4.2 My Applications



View and Apply for Internships: Students see internships filtered by their major and year. They may apply unless the internship is filled/expired or they already have 3 active applications.

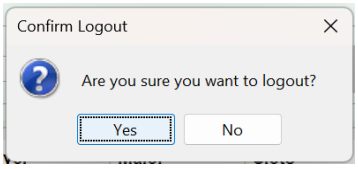
Manage Applications and Placements: Students can view the live status of all their applications.

- **Accept Placement:** If an application is SUCCESSFUL, the student may accept it. Doing so automatically withdraws all other pending applications.

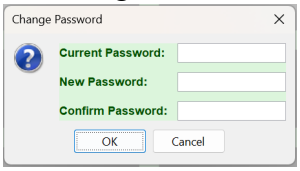
- **Request Withdrawal:** Students may request withdrawal for PENDING applications; these requests go to staff for approval.
- **Post-Confirmation State:** After accepting a placement, students may still browse internships, but applying is fully disabled.

5. Shared User Function

5.1 Log out



5.2 Change Password



- **Log out:** The system provides a function for the user to log out of their account, with a confirmation prompt to prevent accidental session ending.
- **Change Password:** The system provides a feature for any logged-in user to change their password. The user must provide their current password for verification, then enter and confirm a new password. The system validates these inputs before updating the user's credentials.

Figure 7 User Interfaces

5.2 Test Cases and Results

5.2.1 System and Users Functionality

No.	Test Cases	Expected Behavior	Result
1	Valid User Login and Role-Based Access	User should be able to access their dashboard based on their role (Student/Company Rep/Staff) and be redirected to appropriate dashboard	Passed
2	Input Validation	System should check inputs for valid type and value (emails, dates, numbers)	Passed
3	Empty Field Validation	System validates that required fields are not empty before submission	Passed
4	Incorrect Password	System should deny access and alert the user to incorrect password	Passed
5	Password Change Functionality	System updates password and allows login with new credentials	Passed
6	Internship Filter	User should be able to filter internships by level, status, company, and major	Passed

5.2.2 Student Functionality

No.	Test Cases	Expected Behavior	Result
1	Internship Visibility and Application Based on Eligibility	Internships are visible to students based on their year level and major matching	Passed
2	Application Withdrawal Request	Students can submit withdrawal request and is restricted from further actions until processed	Passed
3	Application Status Management	Applications can be viewed with status updates (PENDING, SUCCESSFUL, UNSUCCESSFUL, WITHDRAWN)	Passed
4	Past Deadline Restriction	Students cannot apply to internships past their closing date	Passed
5	Filled Internship Restriction	Students cannot apply to internships that are already filled	Passed
6	Duplicate Application Prevention	System prevents students from applying to the same internship twice	Passed
7	Notification Panel Display	Students see notifications for successful applications and pending withdrawals	Passed

5.2.3 Company Representative Functionality

No.	Test Cases	Expected Behavior	Result
1	Company Rep Registration	System allows new company representatives to register through GUI	Passed
2	Create Internship	Approved reps can create internships with title, description, slots, level, major, and dates	Passed
3	Internship Slot Validation	System validates that slots are between 1-10	Passed
4	Maximum Internship Limit	System prevents creation of more than 5 active internships per company representative	Passed
5	Edit or Delete Pending Internships	Company reps can edit or delete PENDING internships only	Passed
6	Toggle Internship Visibility	Company reps can toggle visibility of their approved internships	Passed
7	View Applications	Company reps can view all applications for their internships	Passed
8	Approve Application	Company reps can approve PENDING applications if slots are available	Passed
9	Reject Application	Company reps can reject PENDING applications with	Passed

		appropriate status update	
10	Application Status Updates	System correctly updates application status to SUCCESSFUL or UNSUCCESSFUL	Passed

5.1.4 Career Center Staff Functionality

No.	Test Cases	Expected Behavior	Result
1	View Company Reps	Staff can view all company representatives	Passed
2	Approve/Reject Company Representative	Staff can approve/reject pending company representatives, changing status to APPROVED/REJECTED	Passed
3	View Internships	Staff can view all internships	Passed
4	Approve/Reject Internship	Staff can approve/reject pending internships with status update	Passed

6. Reflection

6.1 Difficulties Encountered and Solutions

1. Data Persistence Without a Database: implementing Java's Object Serialization, which allowed us to save and load the application's object state directly.

2. Managing Complex Inter-User State: ensuring data integrity when one user's action affected others by using a centralized data manager class (Singleton pattern) to handle all modifications, which kept the system state consistent.

6.2 Knowledge Learnt

1. The Practical Value of Design Patterns: the MVC pattern was essential for separating concerns, while the Singleton pattern ensured a consistent data state.

2. Design Before Implementation: creating thorough UML diagrams first can significantly save refactoring time.

6.3 Suggestions for Improvement

1. Automated Unit Testing: Our manual testing was time-consuming. Implementing a framework like JUnit would allow for faster, more reliable testing and more confident refactoring.

7. GitHub Repository Link

GitHub Repository Link: https://github.com/Phattarachai1/SC2002_SCEA_Group3.git