# **JavaFX Human Resource Management and Payroll System Report**

**Course:** COMP2130

**Assignment:** Assignment 2 (group project)  
  
**Due Date:** December 1, 2024

**Group members:** Mackenzie Bishop, Shalev Haimovitz, Vincente Sequeira, Shineng Zheng

#### **Project Overview**

The Human Resource Management and Payroll System is primarily concerned with efficient management of elementary payrolls and employee information. JavaFX is used as a database structure in building the system employing OOP, which has visible features such as employee registration and data retention capabilities.

#### **Organization and System Architecture**

* + **Main.java**: The application entry point, it launches the JavaFX GUI via EmployeeManager.
  + **EmployeeManager.java**: Handles the GUI setup, the scene transitions, the form submission, and the input validation.
  + **EmployeeData.java**: Manages employee data, which includes file initialization, data serialization, and deserialization.
  + **Resources.java**: Provides paths for resources like the CSS stylesheet, facilitating resource management.

#### **Features and Functionalities**

##### **The employee Registration Form**

* **Fields:** Name, Phone Number, Email, Department, Work Hours, Hourly Rate, and Bonus.
* **Validation Mechanism:**
  + required fields to avoid instances of blank fields.
  + Validation of the email by the use of a regular expression to make sure there is proper formatting.
  + User feedback provided through status labels (with visuals like red text for errors, green for success).

##### **User Interface Design**

* **JavaFX Layouts:**
  + **VBox** and **HBox** containers are used in the organization of form components.
  + **ScrollPane** makes sure the form is accessible regardless of screen size, improving usability.
* **Styling with CSS:**
  + **index.css** is applied dynamically using the Resources class.
  + Custom classes (.error, .success, .title, .card, .container) are used in styling form labels, error messages, and status indicators.

##### **Scene Management**

* The makeFormScene method is used to dynamically create scenes, generating uniform layouts across different screens.

#### **Work Delegation**

Shalev Haimovitz – Lead Developer

Vincente Sequeira – Data Serializer And Developer

Shineng Zheng – UI Developer

Mackenzie Bishop – Developer And Reporter

#### **Code Design and Approaches**

##### **1. The OO Principles**

* **Encapsulation:** Employee-related data is encapsulated within the Employee class, which establishes data consistency and reduces external dependencies.
* **Modularity:** Separating logic into different classes (EmployeeManager, EmployeeData, Resources) improves maintainability and promotes future expansion.
* **Reusability:** Utility methods like makeFormScene and Resources.InitFS avoid code duplication, making the code more readable and easier to maintain.

##### 

##### **Data Persistence with Serialization**

* **Serialization Approach:**
  + Employee data is serialized and stored in employee.data to ensure data persistence between application sessions.
  + For reading and writing objects **ObjectInputStream** and **ObjectOutputStream** are used, which allows seamless deserialization into ArrayList<Employee>.
* **File Initialization:**
  + The Resources.InitFS method creates essential directories and files if they do not exist, making sure the system is ready to operate on first launch.
  + Error handling ensures that file operations do not crash the system, with descriptive error messages guiding users in case of permission issues.

##### **Form Validation and Feedback**

* **Validation Logic:**
  + Input fields will be validated before submission to confirm complete and correct data entry.
  + Invalid email formats are prevented by regular expressions for email validation, enhancing data integrity.
* **User Feedback:**
  + Visual feedback is provided through labels that will change color based on whether it is a success or an error, improving user experience.
  + Status messages update dynamically, guiding the user through the form completion.

##### **Exception Handling**

* Various exception handling techniques are implemented to manage both file I/O and class casting errors:
  + **EOFException Handling:** Will recognize when files are empty and initializes with an empty employee list.
  + **ClassNotFoundException Handling:** Will prevent application crashes from corrupted or incompatible serialized data.
  + **IOException Handling:** Provides informative messages for errors regarding file system, ensuring user awareness of permission issues.

#### **Testing Strategy**

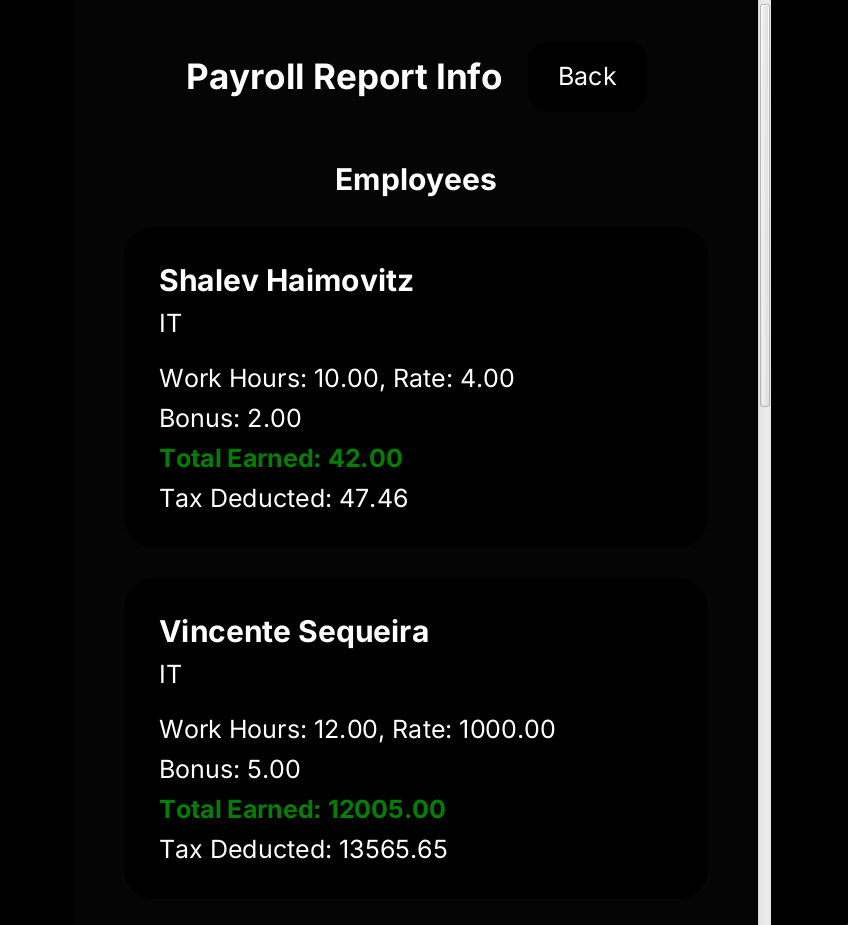
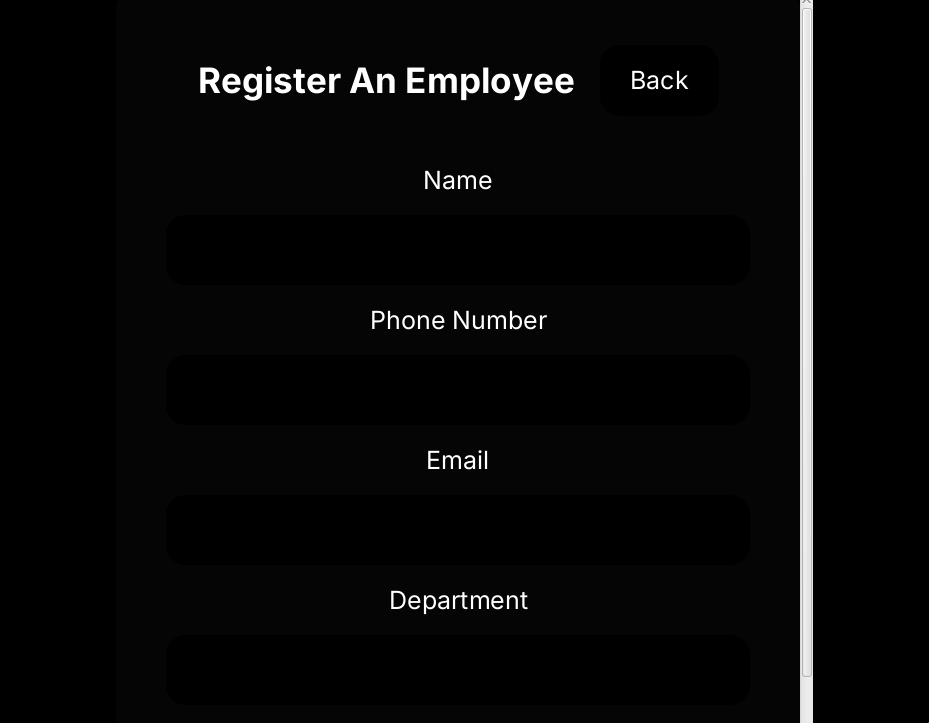
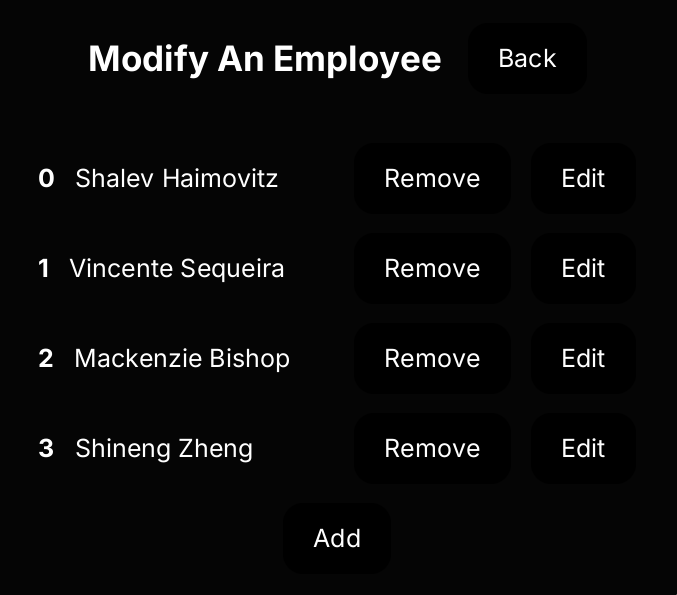
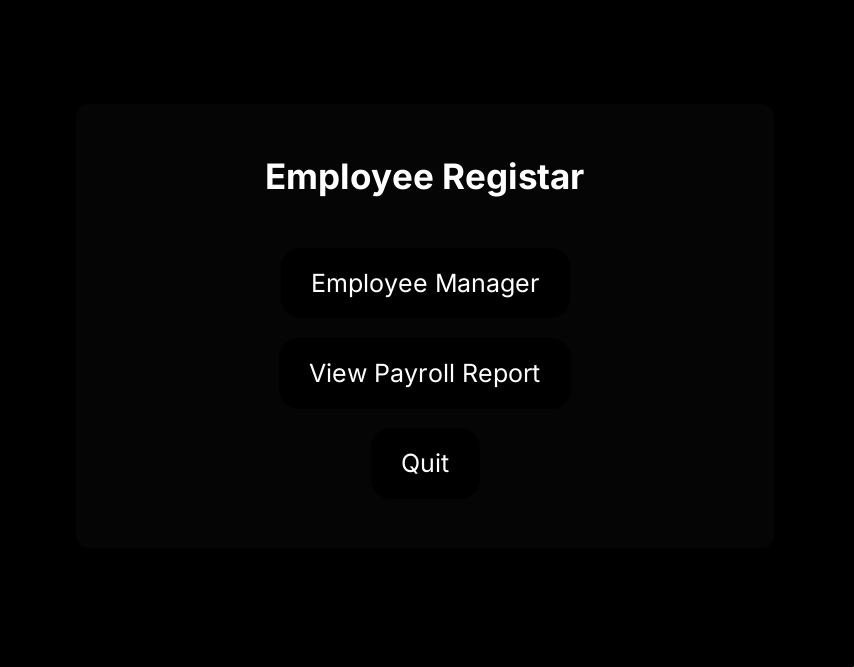
##### **Manual Testing**

* Verified field validations for completeness and correctness (e.g., email formatting, required fields).
* Confirmed scene transitions and layout responsiveness.
* Checked that employee data continued between application restarts by inspecting the employee.data file.

##### **File Handling and Data Integrity Testing**

* Tested scenarios where files were missing or empty, verifying the system handled them gracefully without crashes.
* Checked serialized data consistency by loading and saving multiple employees and verifying the data integrity upon reload.

#### **Usage Demonstration**



#### **Conclusion**

The Human Resource Management and Payroll System is an example of a scalable, well-organized approach to payroll and employee data management. The system provides a dependable and user-friendly experience by integrating JavaFX, OOP principles, and strong error handling.