

4/28/2014

TEAM
SPAM

JUST THE JOB: FINAL REPORT



Sean Gibbens, Phil Dwyer, Adam Sanders, and Max Brodbeck

Table of Contents

<u>Abstract</u>	pg. 3
<u>Chapter 0: Introduction</u>	
Problem Statement	pg. 4-5
Requirements	pg. 6-7
Description of Software Development Process	pg. 8
Team Structure and Roles	pg. 9
Organization of Report	pg. 9
<u>Chapter 1: Project Iteration 1</u>	
<u>Analysis:</u>	
Analysis Domain Model	pg. 10
<u>Requirements:</u>	
Use Case Diagram: All Requirements	pg. 11-12
Use Case Scenarios	pg. 13-16
High-Level Sequence Diagrams	pg. 17-22
CRC Cards	pg. 23-24
High-Level Analysis Class Diagram	pg. 25
Package Diagrams	pg. 25
<u>Design:</u>	
Detailed Class Diagram	pg. 26
Detailed Interaction Diagrams	pg. 27-32
Overall GUI Design	pg. 33-38
Package Diagrams	pg. 39
<u>Implementation:</u>	
Tested Code	pg. 40
Organization using Package Diagrams	pg. 40
<u>Chapter 2: Project Iteration 2</u>	
<u>Analysis:</u>	
Analysis Domain Model	pg. 41
<u>Requirements:</u>	
Use Case Diagram: All Requirements	pg. 42-43
Use Case Scenarios	pg. 44-52
High-Level Sequence Diagrams	pg. 53-66
CRC Cards	pg. 67-69
High-Level Analysis Class Diagram	pg. 70
Package Diagrams	pg. 70
<u>Design:</u>	
Detailed Class Diagram	pg. 71-76
Detailed Interaction Diagrams	pg. 77-90
Overall GUI Design	pg. 91-105
Package Diagrams	pg. 106-107
<u>Implementation:</u>	
Tested Code	pg. 108
Organization using Package Diagrams	pg. 108

Chapter 3: Project Iteration 3

Static Design Model with Modifications	pg. 110-117
Dynamic Design Model with Modifications	pg. 118-129
GUI Model	pg. 130-151
Discussion on Flexibility of Design	pg. 152
Tested Code added to Existing Code	pg. 152

Chapter 4: Conclusions

Views on Software Development and Engineering	pg. 153
Thoughts on Object-oriented Approach	pg. 153

Chapter 5: Team Organization and Roles

Role and Detailed Contribution of Each Member	pg. 154
Overall Work Division and Execution	pg. 154

Appendices

Peer Evaluation Forms	pg. 155-181
Final CRC Cards	pg. 155-177
Final Use-case Diagram	pg. 178-180
	pg. 181

Abstract

In this project, we were assigned to work on the Just the Job project description located in our textbook. Throughout the semester this spring, we have worked on three iterations to produce the software described in the requirements of our project. With this process, we overcame issues involving learning new subjects and scheduling with team members. As we progressed through our education of this class, we developed multiple static and dynamic design diagrams that correlated to our actual code for the software. These diagrams helped our group map and develop our code by defining our objects in entities, controllers, and boundaries. With our functionality working for each iteration, we looked into improving our GUI design. This is very noticeable in the transition from iteration one to iteration two. To complete our analysis, we looked at the back-end of our software, which we continued to use common separated value (CSV) text files instead of using an Access database back-end. As a group, we wanted this software to have an easier transition to a new back-end source. In the end, as a group we feel that our software is very functional and covers the requirements stated by the project description for Just the Job.

Chapter 0: Introduction

Problem Statement:

Just the Job is a company that provides house cleaning services on a one-off basis, for example when people move house.

At the moment, when a potential customer contacts the Just the Job office, the receptionist books an appointment for the office manager to visit the property to be cleaned and give the customer a date and price for the job. Once these have been agreed, a booking form is filled out; one copy of the form is given to the customer and two copies are filed at the Just the Job office.

On the date arranged, a team of two or three cleaners arrive at the property and carry out the cleaning as specified. The customer then signs a copy of the original booking form to confirm the job has been carried out satisfactorily. When the signed booking form arrives back at the Just the Job office, the receptionist sends an invoice to the customer for the payment. A receipted copy of the invoice is sent to the customer when full payment is received.

Just the Job also deals with customers who require cleaning services on a regular basis. This cleaning is carried out on the same day each week, and is charged at an hourly rate, negotiated with the customer. The office manager tries to send the same cleaner each week, as this helps customer relations.

Just the Job allocates customer numbers and keeps details on file of all its customers for marketing purposes. The office also keeps records of all the cleaners, including name, address, contact number and the number of hours worked each week.

The office manager has decided that she needs a new computer system to handle most of the paperwork involved in Just the Job's daily routines.

The new system must keep a record of customers, cleaners and jobs. The office manager, Eileen, wants to be able to use the system to produce printed monthly invoices for regular customers and one-off invoices for single jobs. She would also like the system to produce a weekly schedule for each cleaner showing where and when they are working. This will be given to the cleaners at the start of the week along with a copy of the Booking Form for the customer to complete. The system will also be used to produce a weekly list showing how many hours each cleaner has worked.

Invoices for one-off jobs are to be printed and sent out as soon as the signed booking form is returned to the office. Invoices for regular jobs are to be printed and sent out once a month. Customers who have regular cleaning jobs on several properties should receive a single invoice.

Eileen would also like the system to be able to keep track of her appointments and produce a printed schedule for her.

Requirements

1. Manage customers (R & M)

- a. Add a new customer (R)
- b. Delete an existing customer (M)
- c. Modify information stored about an existing customer (R)
- d. Display/print information about an existing customer based on his/her id or name (R)
- e. Display/print a list of all customers and their information (R)

2. Manage employee (i.e. cleaner) (M & R)

- a. Add a new employee—name, id, address, pay-rate, weekly schedule, etc. (R)
- b. Delete an existing employee (M)
- c. Display/print information about an existing employee (R)
- d. Display/print a list of all employees with their information (R)
- e. Modify information stored about an existing employee (R)

3. Manage jobs/appointments (M & R)

- a. Add/register a new job (R)
- b. Delete/cancel an existing job (M)
- c. Display/print the status of a specific job (e.g. job number; address of property, completion status, by-whom etc.) (R)
- d. Display/print a list of all jobs with their completion status (R)
- e. Display/print a list of all completed jobs (R)
- f. Display/print a list of all pending jobs (R)
- g. Modify information stored about an existing job (R)

4. Manage customer invoices for one-time jobs (R & M)

- a. Create a new invoice (R)
- b. Cancel an existing invoice (M)
- c. Modify an existing invoice (R)
- d. Display/print an existing invoice (R)
- e. Display/print a list of all invoices and their payment status (R)

5. Manage customer invoices for regular jobs (R & M)

- a. Create a new invoice (R)
- b. Cancel an existing invoice (M)
- c. Modify an existing invoice (R)
- d. Display/print an existing invoice (R)
- e. Display/print a list of all invoices for a given week with payment status (R)

6. Manage customer payments (R & M)

- a. Record a full payment (R)
- b. Cancel/credit a payment (M)
- c. Print receipt (R)

7. Maintain manager's personal weekly schedule (R)

- a. Add a new appointment (R)
- b. Cancel an existing appointment (R)
- c. Modify an existing appointment (R)
- d. Print weekly schedule (R)

8. Save information (R & M)

Upon user request, save all data to disk at any time. (This is in addition to the automatic save to disk which occurs at shutdown)

Description of Software Development Process

We developed our software using the Agile Software Development with three separate iterations. The first iteration developed basic functionality with only a select number of basic requirements like add customers, employees, and jobs. This was helpful for our group to handle as these use cases were the foundation of the software as other entities and use cases branched from the three basic entities. Also with the first iteration, we basically disregard the GUI as we rated functionality more important as the front-end. The GUI can also be redone after functionality is completed. After our first presentation to the class, we reviewed our diagrams and code with new requirements to add on top of our work. This was very helpful in that we could diagnose our problems and find solutions before our software and documentation became too voluminous.

In our second iteration, we added more specific use cases to our basic entities and developed our GUI to a more recognizable version of what we have now. In our GUI during this iteration, we added list boxes and buttons with multiple windows including a log-in form. We also developed our back-end by using our object entities and write to common separated value (CSV) text files. This was completed during the second iteration as we give our three basic entities all of their use cases and we could develop a standard save-to-file process. After this work was completed, we had another presentation to get outside opinions. With these opinions, we consider the criticism and revised our code and documentation.

In the third iteration, we were asked to branch out with new use cases and entities. Just like the other two iterations, we reviewed and redeveloped our documentation and code. By working on chunks of the requirements, we were able to focus on certain tasks and correctly develop our software. This process also helped us develop a working foundation that was very simple to add on to our software.

Team Structure and Roles

Our team had a very balanced work format with all four members having the same experiences in programming. We did not need a leader for our team as everyone held each other accountable for their parts of the project. Overall, our team worked very efficiently and was able to find time in our busy schedules to meet as a team to get the work needed completed.

Roles

Sean Gibbens – Diagram Lead

Phil Dwyer – Team/Documentation Lead

Adam Sanders – Code Lead

Max Brodbeck – The Dabbler (worked on everything including diagrams and code)

Organization of Report

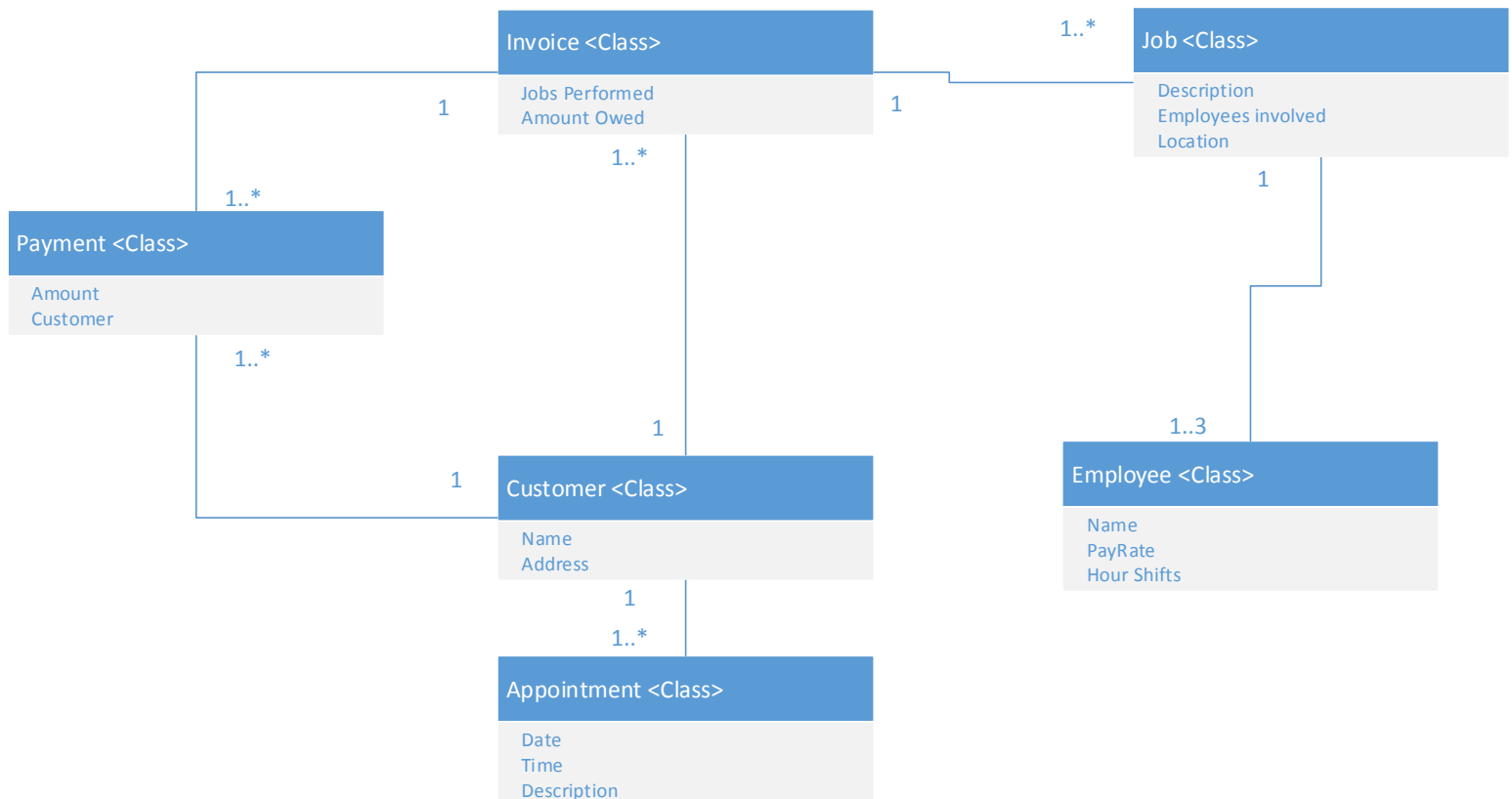
In this report, you will encounter our understanding and development of Unified Modelling Language (UML) diagrams and the programming language of VB.NET. In the first chapter, the report describes our first iteration and shows our output from this iteration, which includes our first diagrams and code. In the second and third chapters, the report describes the other two iterations and displays our improvements on our diagrams and code from criticism received during presentations in class. The fourth chapter sums up our thoughts on the project and software engineering as a whole. In the fifth chapter, the report will explain our more detailed roles and how we executed these roles as members of Team SPAM. For all of other documentation including peer evaluation forms please refer to the Appendices at the end of this report.

Chapter 1: Project Iteration 1

Analysis

Analysis Domain Model

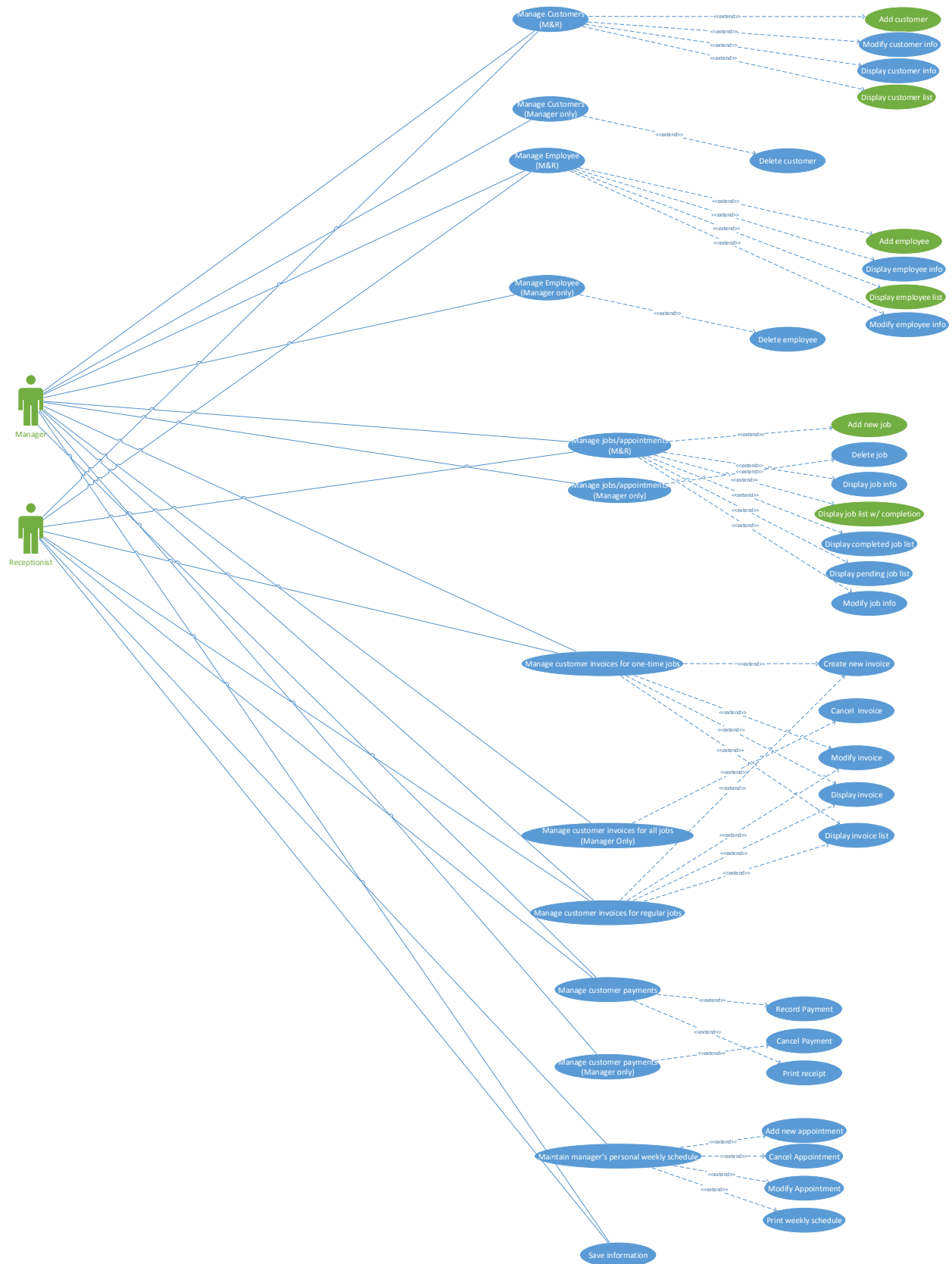
Our analysis domain model for Iteration 1 was the most high-level diagram we produced, as it attempts to model all of the classes in the problem domain in one go. We created this first domain model based upon the noun analysis and formation of candidate classes. Each conceptual entity class has a relationship with the other classes, and these connections are meant to outline the structure for the future code.



Requirements

Use Case Diagram

Note: In this use-case diagram, we covered all requirements for the project. This diagram contains all completed use cases in green ovals. These completed use cases were our tasks we focused on Iteration 1. We did not implement the generalization arrow for this iteration as we were inexperienced with use-case diagrams. The diagram is located on next page. To view the diagram at closer look, refer to our wiki page on Moodle.



Use Case Scenarios

Use-case: Add a new customer (1.a)

Actor(s): Manager, Receptionist

Goal: To have a manager or receptionist add a new customer into the system

Overview: The manager or receptionist accesses the system. They then enter the “add” section of the system. Then they go to “add customer” and enter all of the necessary customer info.

Typical course of events:

Actor Action

System Response

1. Manager or receptionist accesses system
2. Goes to “add” section
3. Goes to “add customer” section
5. Fills out all necessary information and submits

4. Displays “add customer” form
6. Verifies all info is filled out
7. Enters customer info system

Alternative courses:

2. User goes to the wrong section. Would have to go back a page and try again. (*alternate*)
3. User goes to “add job” or “add employee” instead of “add customer”. Requires the user to go back a page and try again. (*alternate*)
5. User enters some wrong information about the customer. This will be caught by system in step 6. (*alternate*)
5. User forgets to submit form. (*alternate*)

Use-case: Display a list of all customers and their information (1.e)

Actor(s): Manager, Receptionist

Goal: To have a manager or receptionist display a complete list of customers and their information

Overview: The manager or receptionist accesses the system. They enter the “customer” section of the system. They then select the “display all” link on this page.

Typical course of events:

Actor Action

System Response

1. Manager or receptionist accesses system
2. Goes to “customer” section
3. Clicks on “display all” link

4. Displays list of all customers and info

Alternative courses:

2. User goes to the wrong section. Would have to go back and try again. (*alternate*)
3. User clicks on the wrong link. Would have to go back and try again. (*alternate*)

Use Case: Add a new employee (2.a)

Actor(s): Receptionist, Manager

Goal: Enter a new employees information into the system. This information includes the employees name, id, address, weekly schedule, pay-rate, etc.

Overview: The Receptionist or Manager accesses the system. They select the 'add' section. They then select the new employee section, and fill out the form boxes to enter in information about the employee.

Typical Course of Events:

Actor's Action

System Response

1. The Receptionist or Manger goes to the 'add' section.
2. They should then go to the 'add new employee' section.
3. They then put that employees information into the correct sections, and submit it.
4. Verify that all the information has been entered
And the correct number of digits has been entered
correctly, i.e. for a phone number.
The information is saved after it is verified.

Alternatives:

Step 1. They choose the wrong section, and have to return to the home page.

Step 2. They choose a different addition section, such as customer, and will have to return to the add page.

Step 3,4. They enter information in the wrong field, for example if they were to put an id number into the name section, the system should catch clearly incorrect information such as this.

Step 3,4. They forget to hit submit, and will have to reenter the information.

Use Case: Display a list of all employees with their information (2.d)

Actor(s): Receptionist, Manager

Goal: A list of all employees, including their information, is displayed on the screen.

Overview: The Receptionist or Manager goes to the employee section. They choose the 'display all' link, and a list of all employees with their information is displayed.

Typical Course of Events:

Actor's Actions

1. The Receptionist or Manager goes to the 'employee' section
2. They then select the 'display all' link.

System Response

3. The system displays a list of all employees with their information.

Alternative:

Step 1 They choose the wrong section, and have to return to the home page.

Step 2 They choose the wrong link, and must return to the employee section.

Use Case: Add/Register a new job (3.a)

Actor(s): Receptionist, Manager

Goal: Enter a job into the system so that employees can be dispatched to it.

Overview: The Receptionist or Manager accesses the system. They select the 'add' section. They then select the 'add new job' section. After that they are taken to a page where they can enter information such as location and description.

Typical Course of Events:

Actor's Action

1. The Receptionist or Manager goes to the 'add' section.
2. They should then go to the 'add new job' section.
3. They then put the proper information about the job into the System, and then hit submit.

System Response

4. The information is saved.

Alternative:

Step 1. They choose the wrong section, and have to return to the home page.

Step 2. They choose a different addition section, such as customer, and will have to return to the add page.

Step 3,4. They forget to hit submit, and will have to reenter the information.

Use Case: Display a list of all jobs with their information (3.d)

Actor(s): Receptionist, Manager

Goal: A list of all jobs, and their completion status, is displayed on the screen.

Overview: The Receptionist or Manager goes to the 'jobs' section. They choose the 'display all' link, and a list of all jobs with their completion status is displayed.

Typical Course of Events:

Actor Action

System Response

1. The Receptionist or Manager goes to the 'jobs' section
2. They then select the 'display all' link.

3. The system displays a list of all jobs with their completion status.

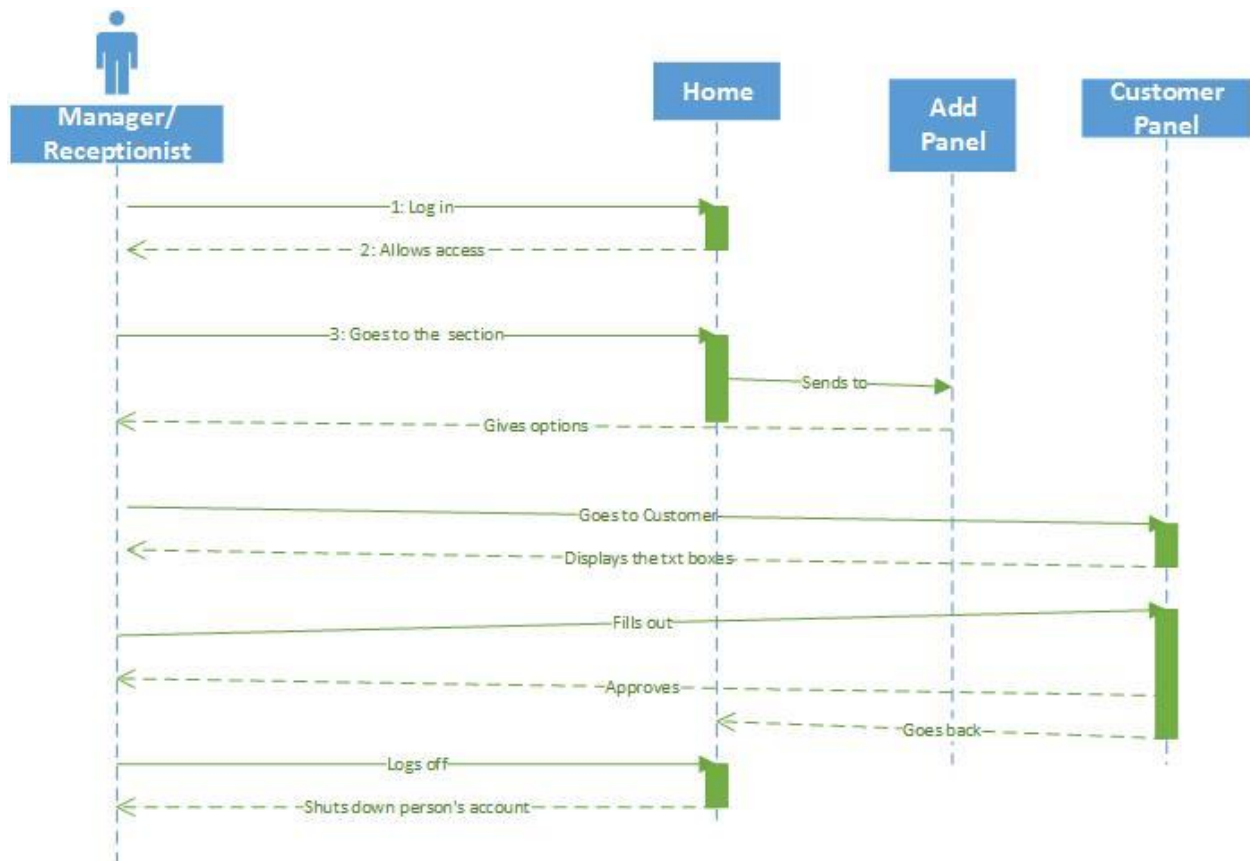
Alternative:

Step 1. They choose the wrong section, and have to return to the home page.

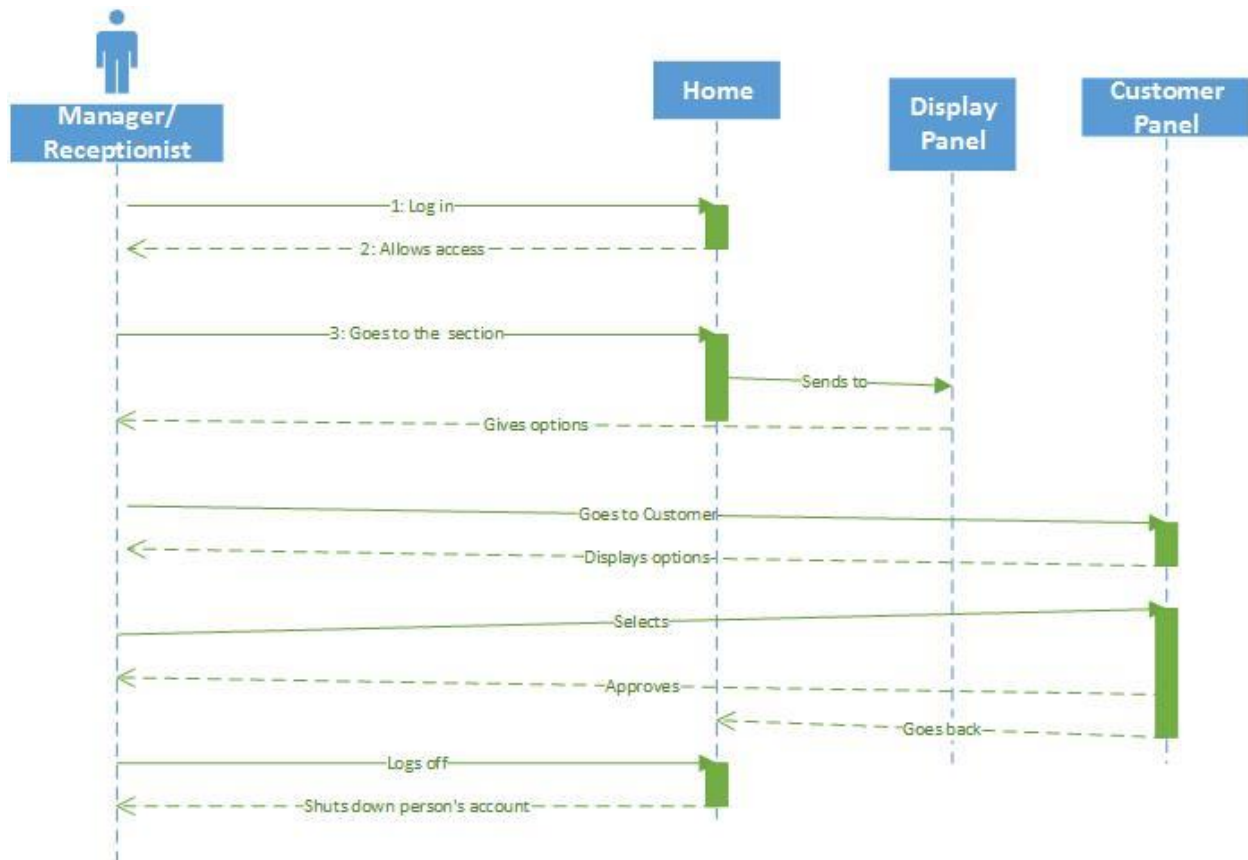
Step 2. They choose the wrong link, and must return to the employee section.

High Level Sequence Diagrams

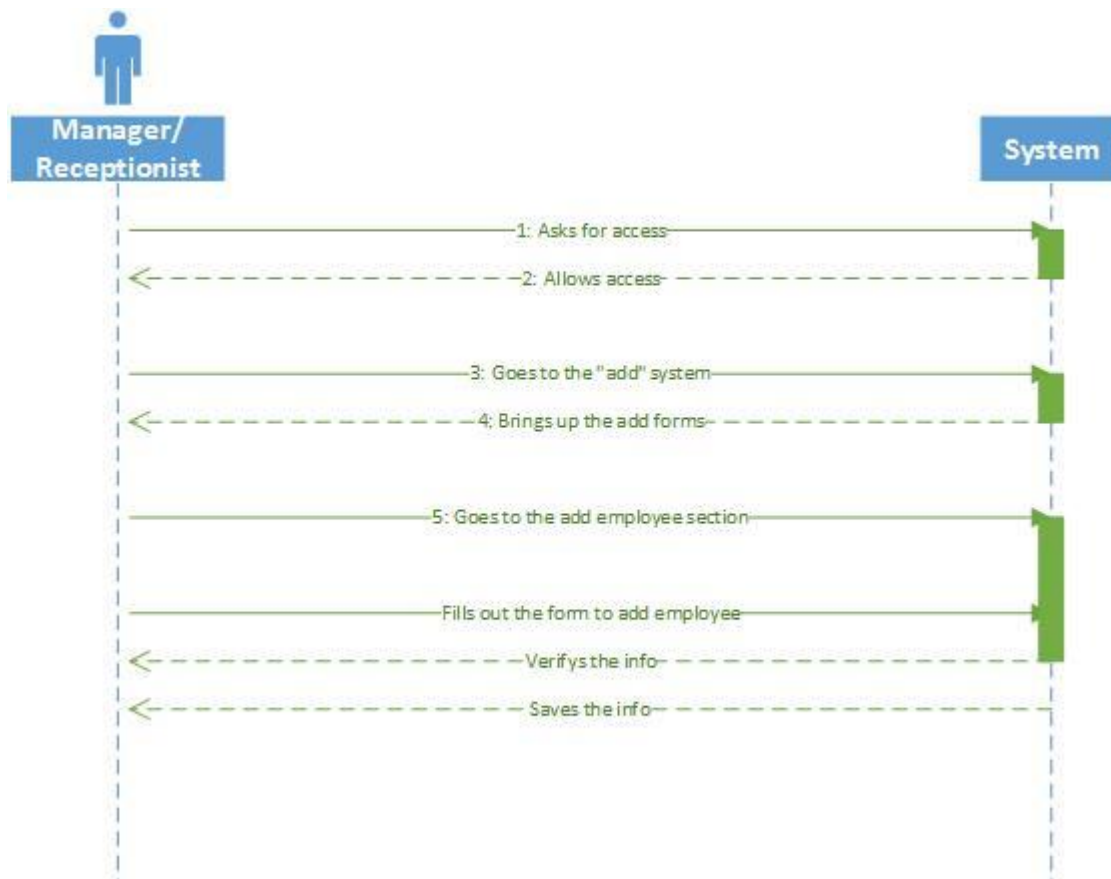
This is the High-level Sequence Diagram for Requirement 1a) Add a new customer



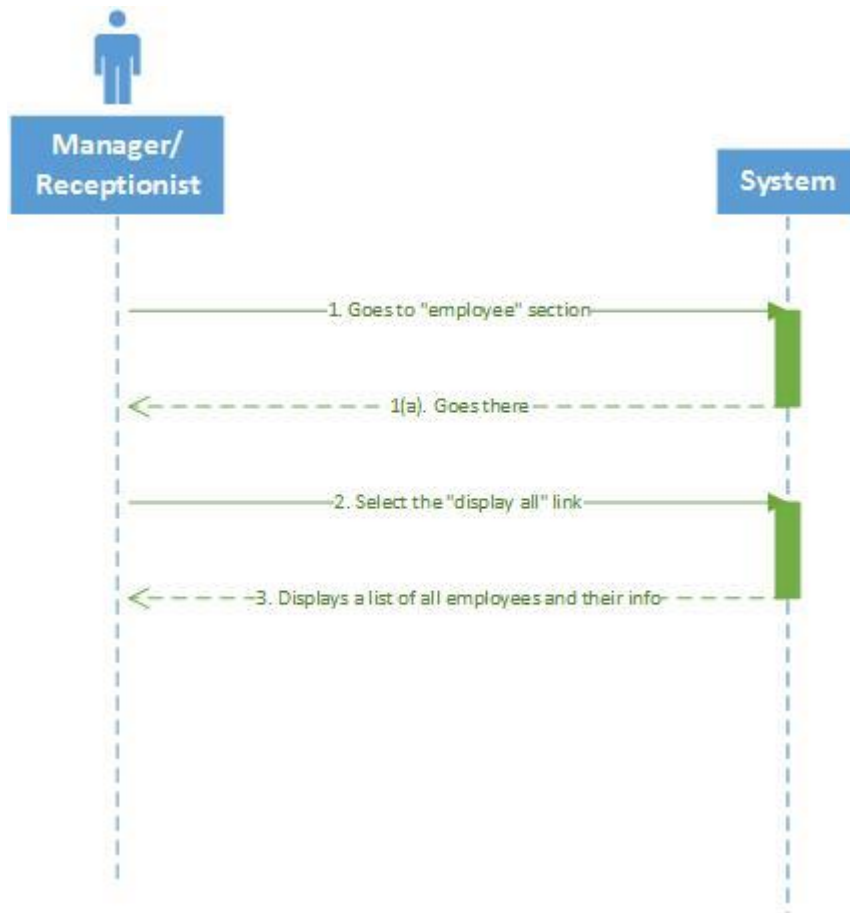
This is the High-level Sequence Diagram for Requirement 1e) Display/print information about an existing customer based on his/her id or name



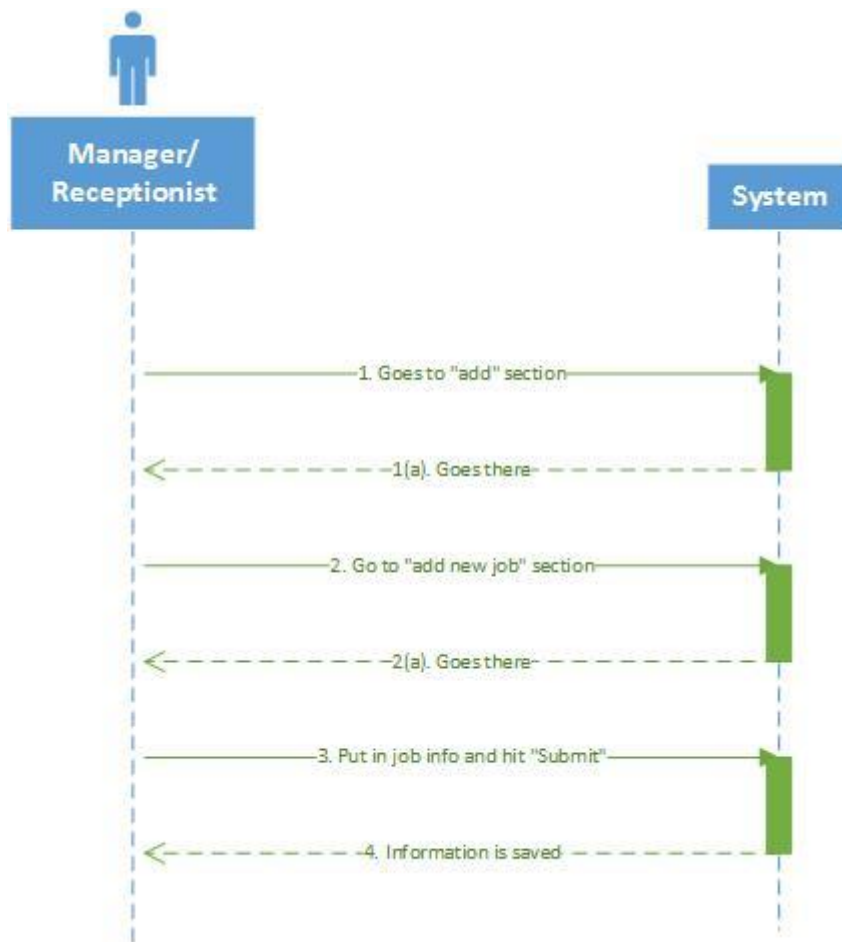
This is the High-level Sequence Diagram for Requirement 2a) Add a new employee



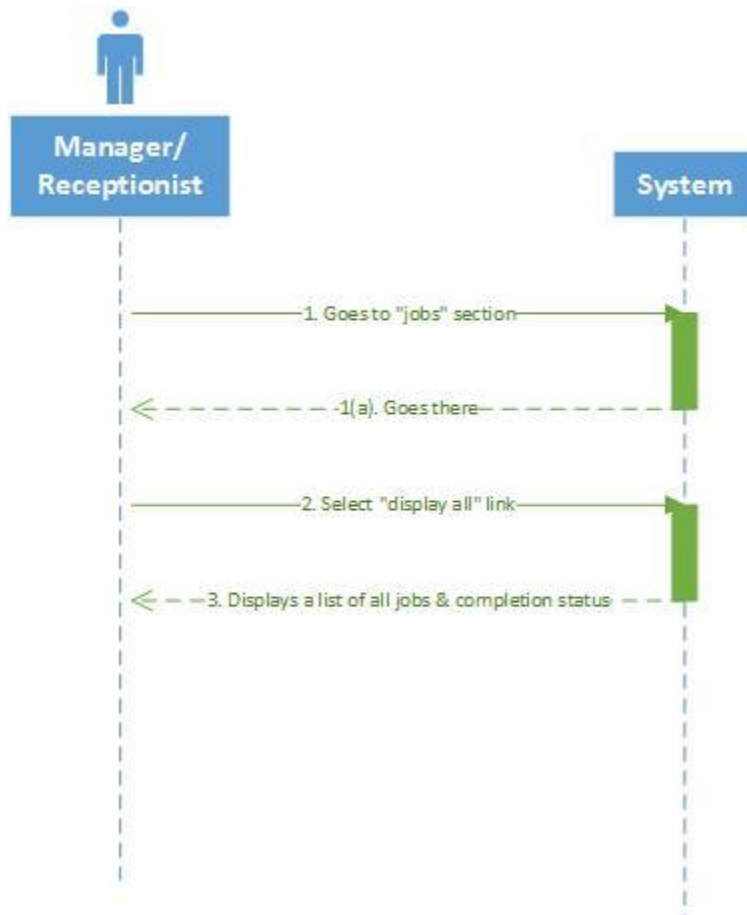
This is the High-level Sequence Diagram for Requirement 2d) Display/print a list of all employees with their information



This is the High-level Sequence Diagram for Requirement 3a) Add a new job



This is the High-level Sequence Diagram for Requirement 3d) Display/print a list of all jobs with their completion status



CRC Cards

Note:

We did not use entity classes in our first iteration, so the only classes that were implemented were boundary and controller classes that wrote directly to the appropriate text file. The following CRC Cards are our Iteration 1 Controller classes.

Add Customer <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Customer to an appropriate text file with customer attributes like ID, Address, and Name of Customer.	

Display Customers <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Display all customers that were added to the text file by the Add Customer Controller.	

Add Employee <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add employee to an appropriate text file with employee attributes like hours for the week, ID, and pay rate.	

Display Employee <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Display all employees that were added to the text file by the Add Employee Controller.	

Add Job <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Job to an appropriate text file with job attributes like ID, Date, and Description of Job.	

Display Job <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Display all jobs that were added to the text file by the Add Job Controller.	

High-Level Analysis Class Diagram

At this point in Iteration 1, we did not develop a full analysis class diagram that included all necessary boundaries, controllers, and entities. For a high-level class diagram modeling the fully functioning system, covering all requirements, please see our Iteration 1 domain model.

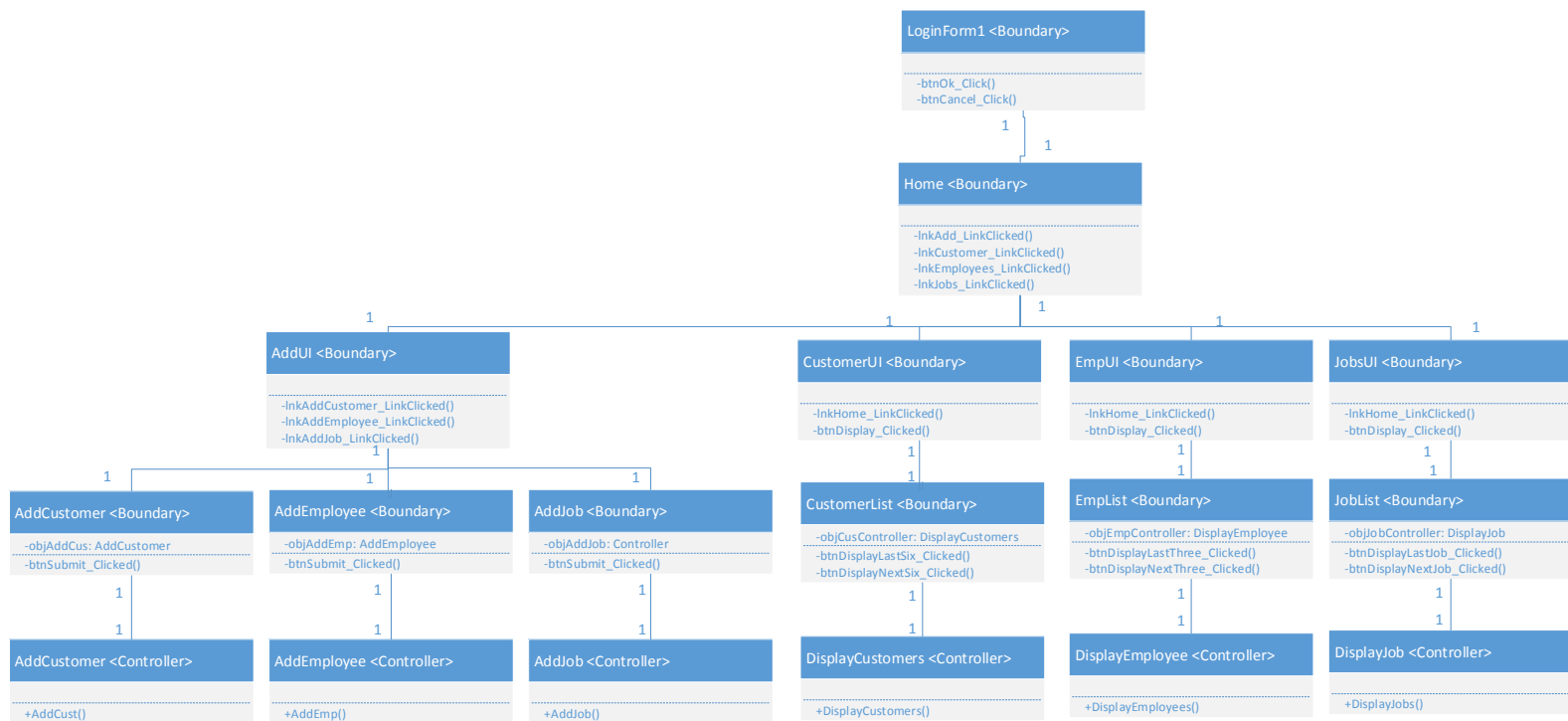
Package Diagrams for Requirements

Because we did not have a complete analysis class diagram at this point in Iteration 1, we could not organize all of our classes in package form. For a package diagram covering all of the requirements from Iteration 1, please see our Iteration 1 design package diagram.

Design

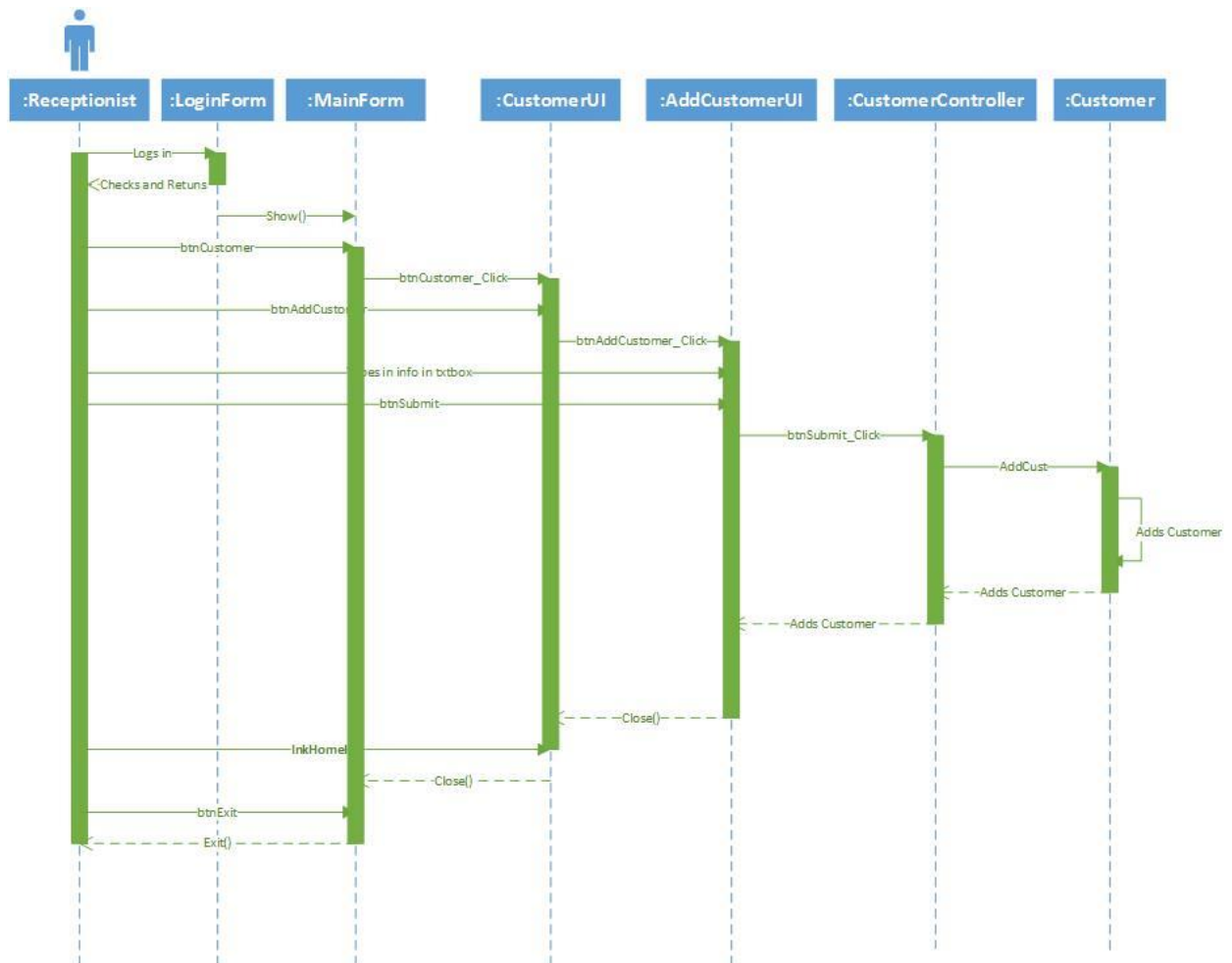
Detailed Design Class Diagram

Our detailed class diagram for Iteration 1 is two-layered. The first layer is composed of our boundary classes, each of which has links to the corresponding forms. The bottom-most layer of boundaries has buttons which trigger the controllers. Notice the separate controller classes used for each individual function, as we consolidate these in future iterations. For a closer look at this diagram, please see our Wiki.

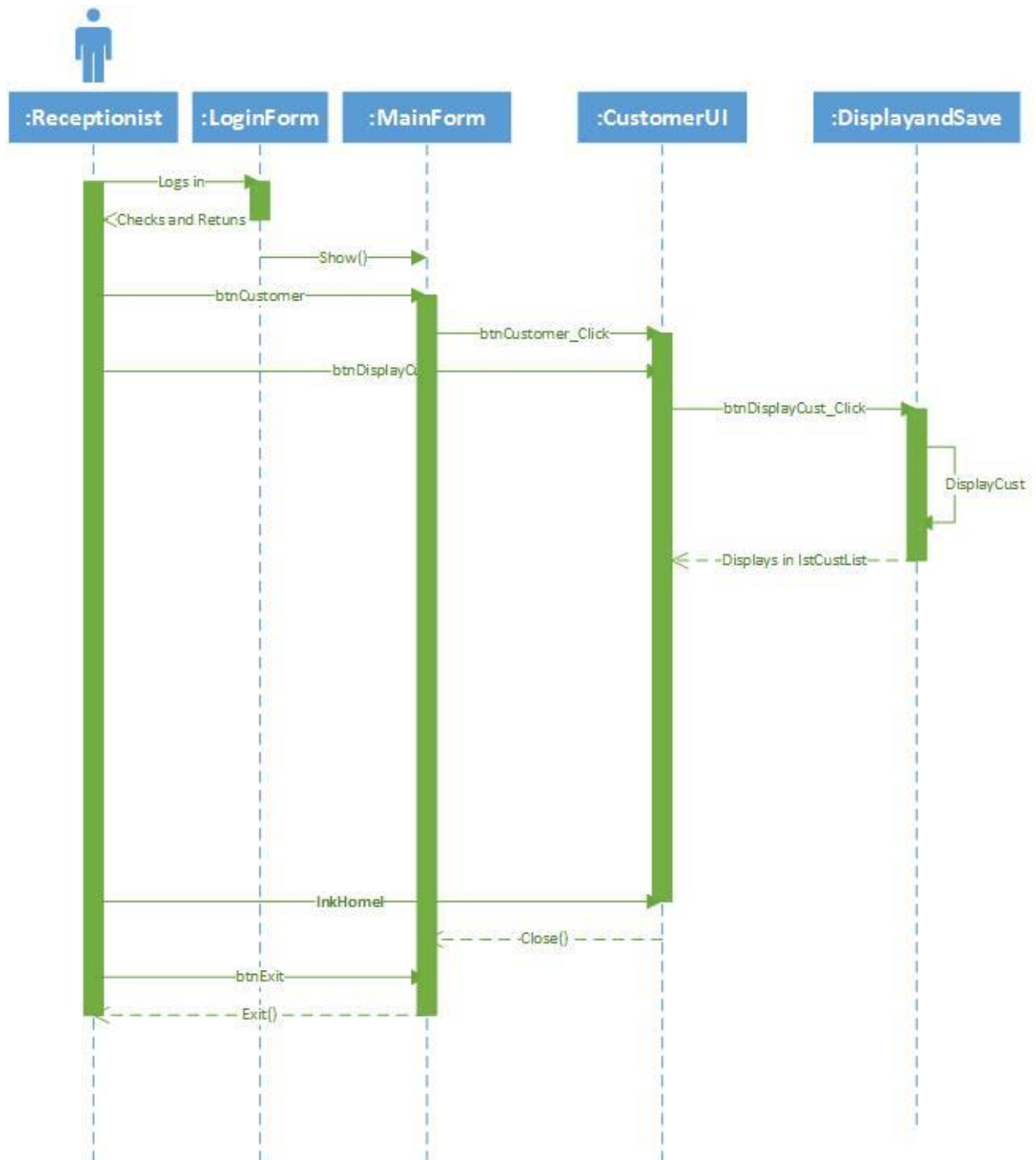


Detailed Interaction Diagrams

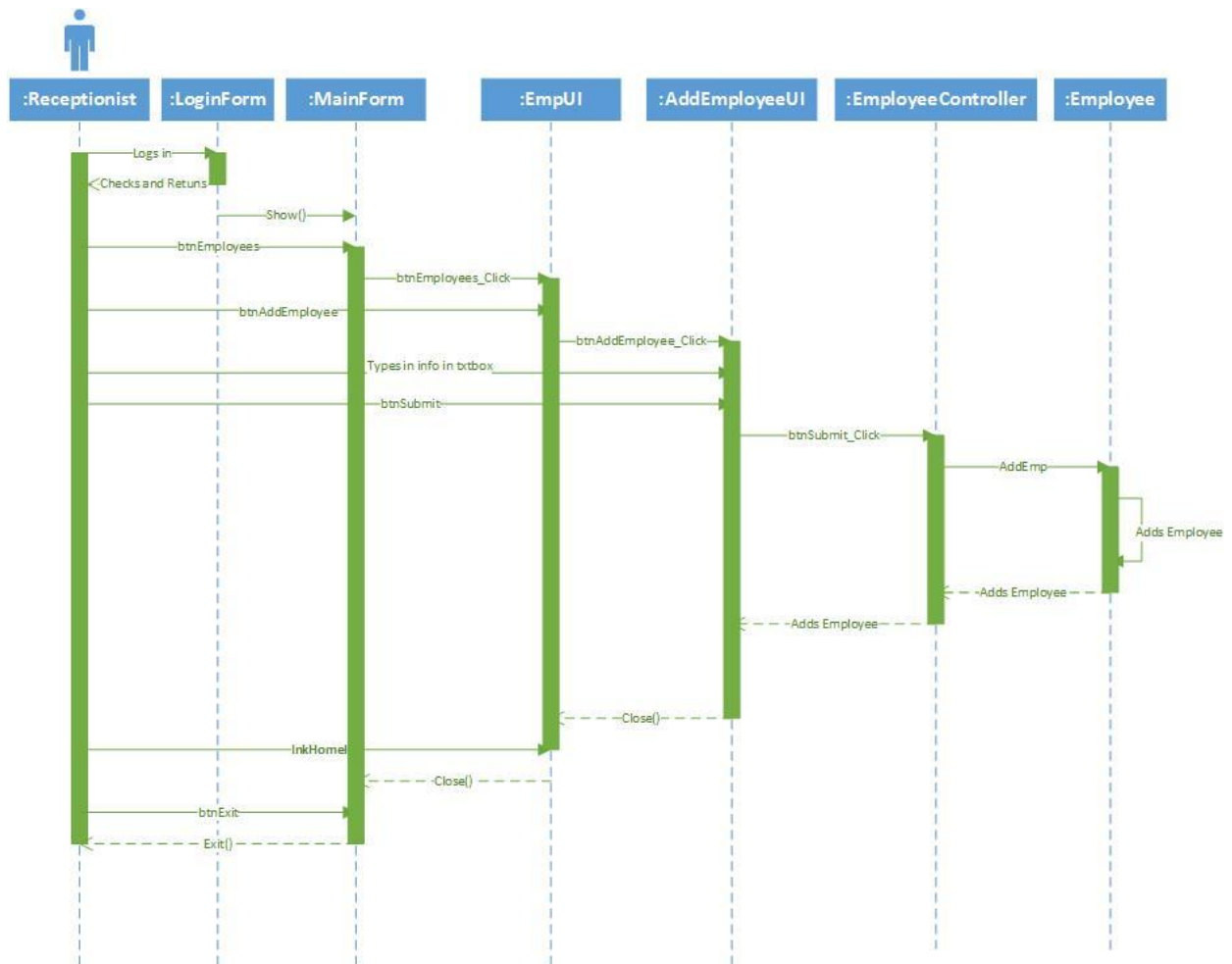
This is a detailed sequence diagram for 1a) Add a new customer



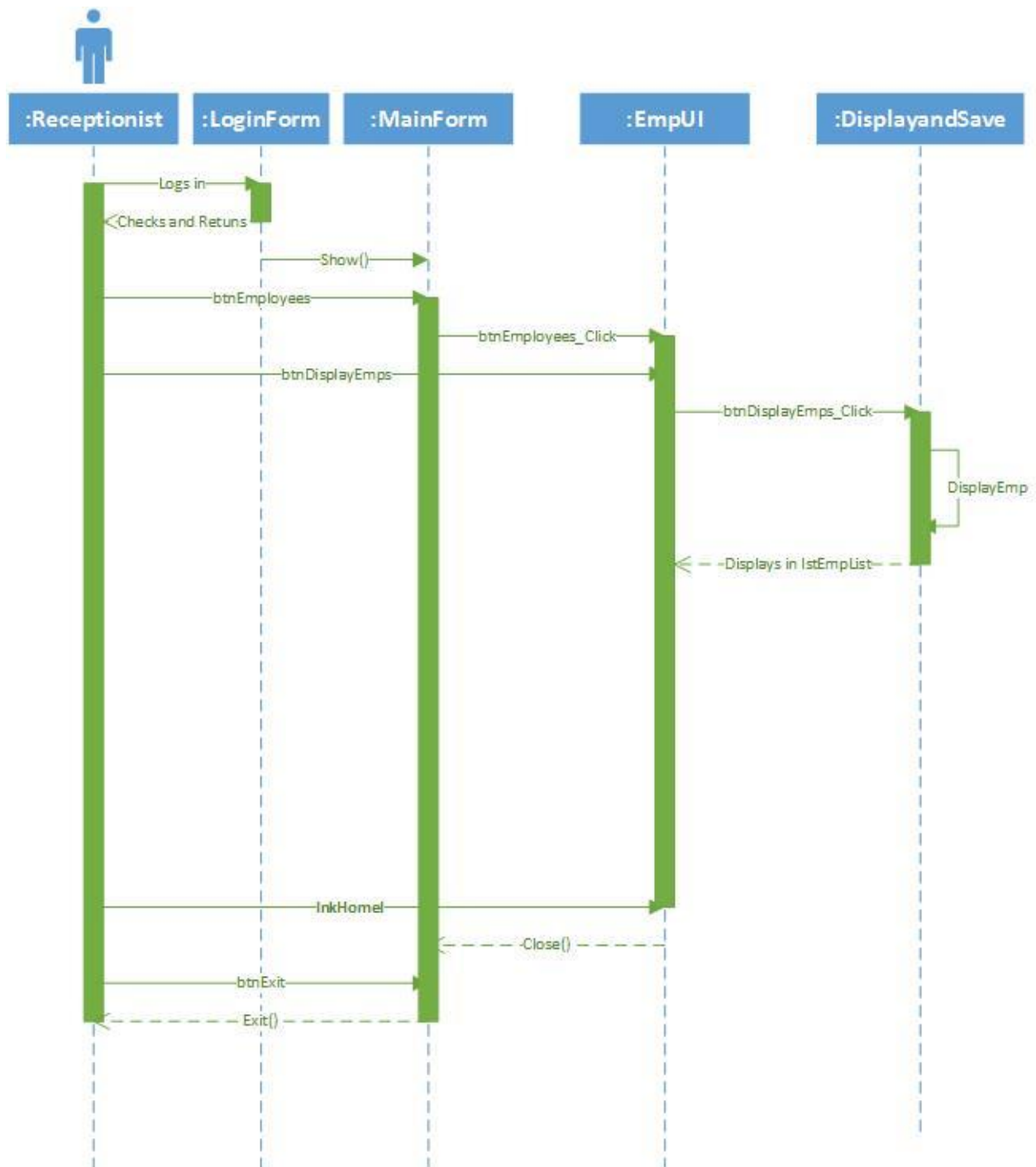
This is a detailed sequence diagram for 1e) Display/print information about an existing customer based on his/her id or name



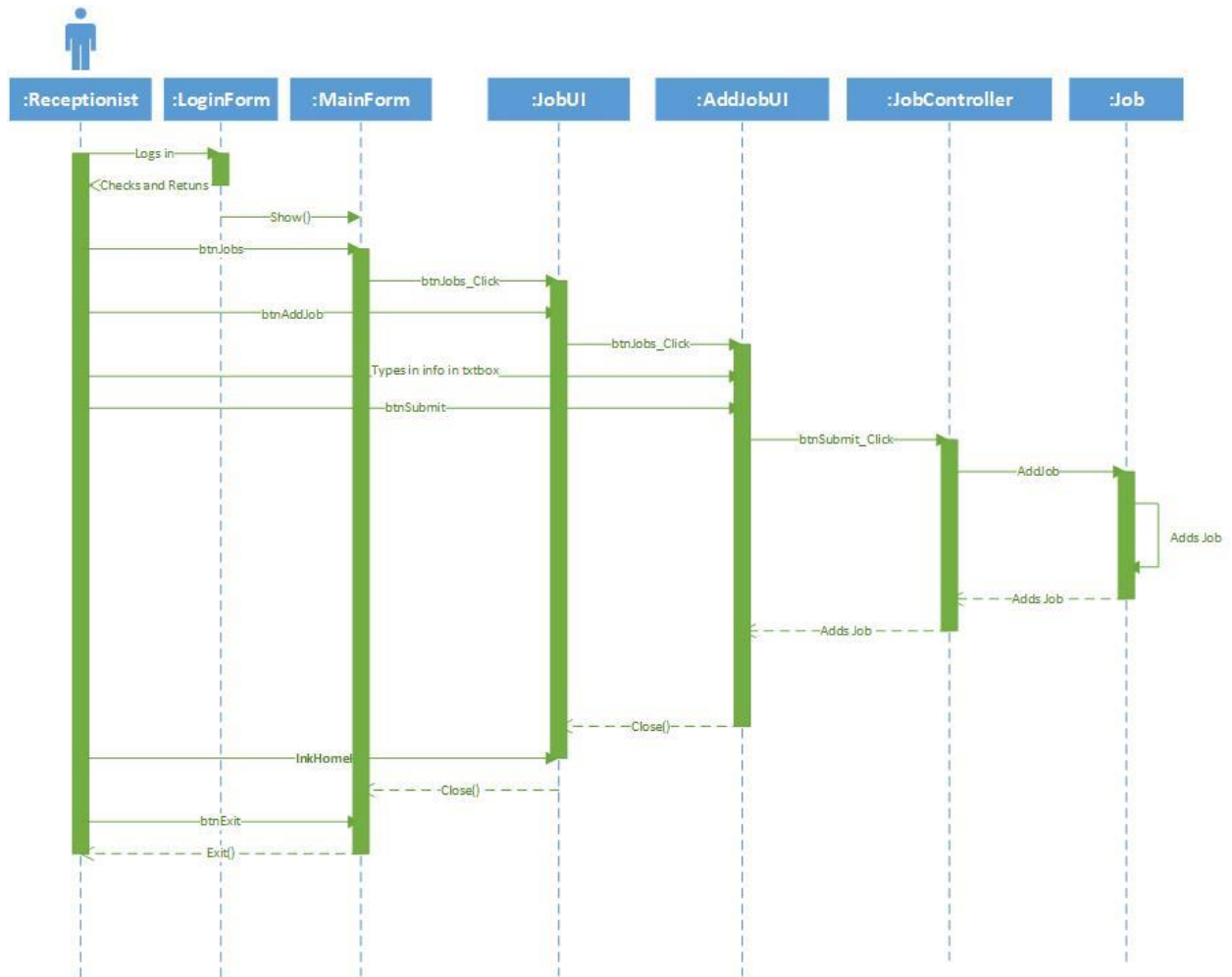
This is a detailed sequence diagram for 2a) Add a new employee



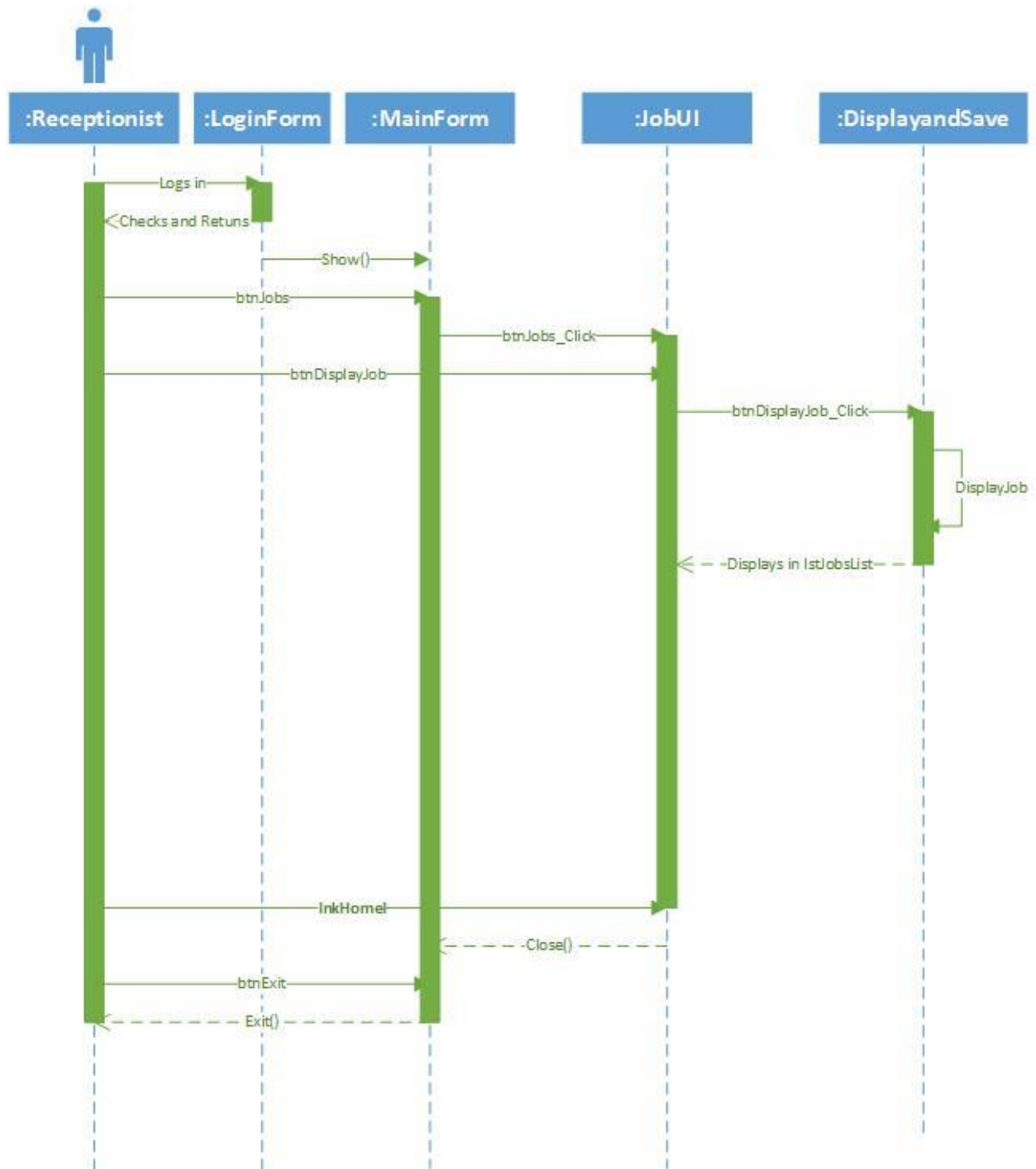
This is a detailed sequence diagram for 2d) Display/print information about an existing employee



This is a detailed sequence diagram for 3a) Add a new job

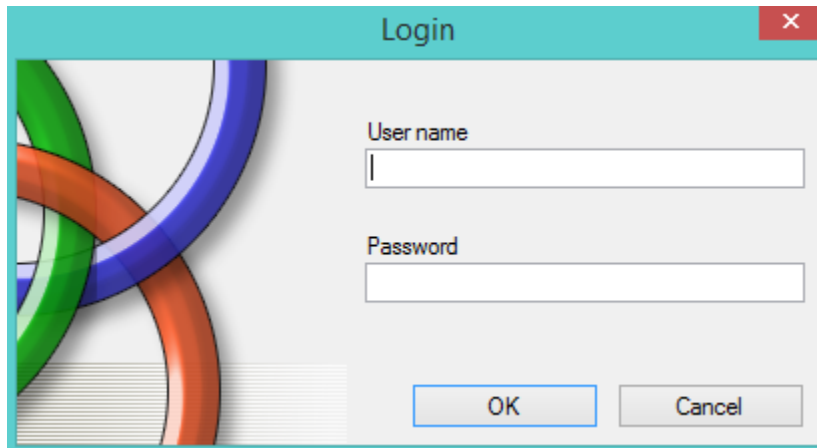


This is a detailed sequence diagram for 3d) Display/print a list of all jobs with their completion status



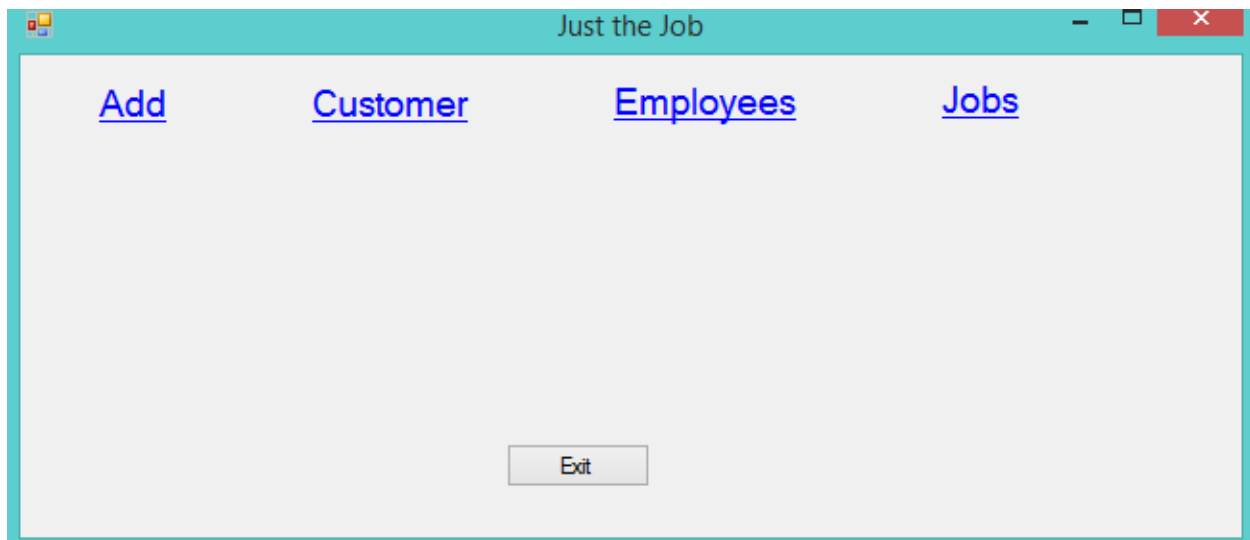
Overall Design for GUI

This is where a user logs in in



Home Menu

This is where a user navigates the system



Add Page

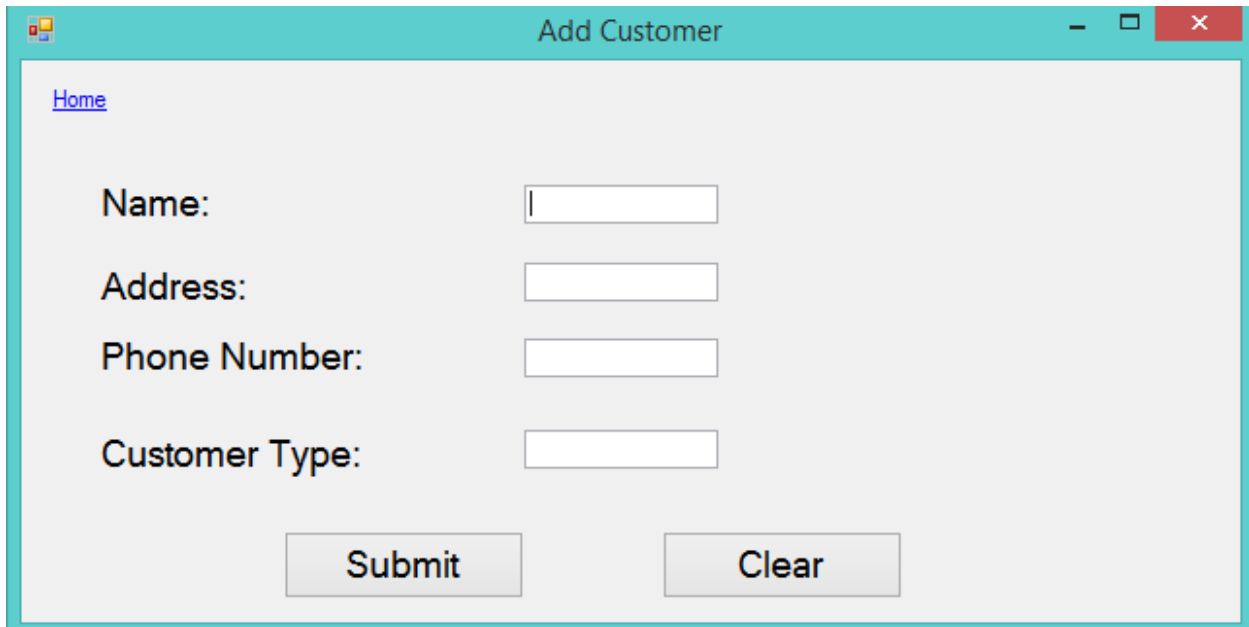
A user would go here to add anything



The screenshot shows a web browser window with the title "Just the Job". The page has a light gray background. At the top left, there is a link labeled "Home". Below it, there are three more links, each on a new line: "Add Customer", "Add Employee", and "Add Job". All links are underlined and in blue text.

Add Customer Page

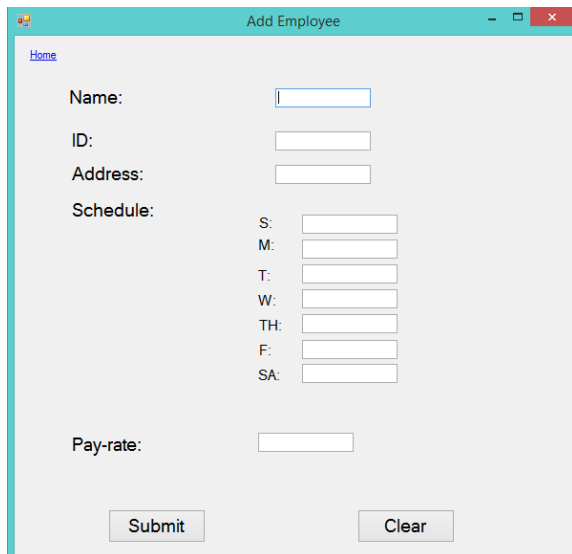
This is the form where you add a customer



The screenshot shows a web browser window with the title "Add Customer". The page has a light gray background. At the top left, there is a link labeled "Home". Below it, there are four form fields, each with a label to its left: "Name:", "Address:", "Phone Number:", and "Customer Type:". Each label is followed by a white rectangular input box. At the bottom of the form, there are two buttons: "Submit" and "Clear".

Add Employee Page

This is the form where a user adds an employee



The 'Add Employee' form is displayed within a browser window. It features a teal header bar with the title 'Add Employee' and standard window controls. A blue 'Home' link is located in the top left corner. The form contains several input fields: 'Name:', 'ID:', 'Address:', and 'Schedule:'. The 'Schedule:' field is a vertical stack of seven inputs labeled 'S:', 'M:', 'T:', 'W:', 'TH:', 'F:', and 'SA:'. Below these is a 'Pay-rate:' input field. At the bottom, there are 'Submit' and 'Clear' buttons.

Home

Name:

ID:

Address:

Schedule:

S:

M:

T:

W:

TH:

F:

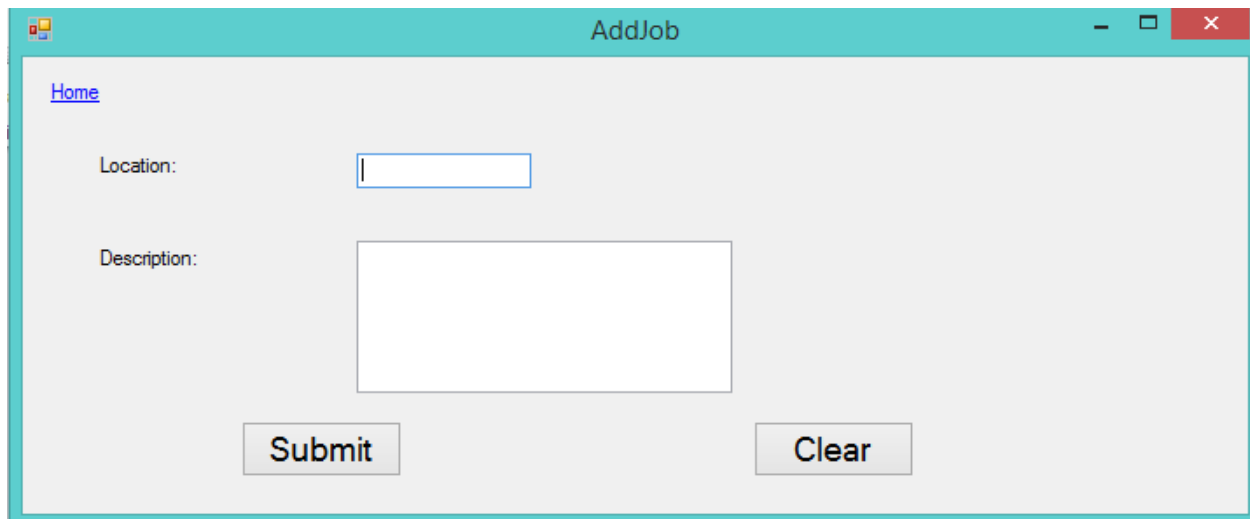
SA:

Pay-rate:

Submit Clear

Add Job Page

This is the form where a user adds a job



The 'AddJob' form is displayed within a browser window. It features a teal header bar with the title 'AddJob' and standard window controls. A blue 'Home' link is located in the top left corner. The form contains two input fields: 'Location:' and 'Description:'. The 'Description:' field is a large text area. At the bottom, there are 'Submit' and 'Clear' buttons.

Home

Location:

Description:

Submit Clear

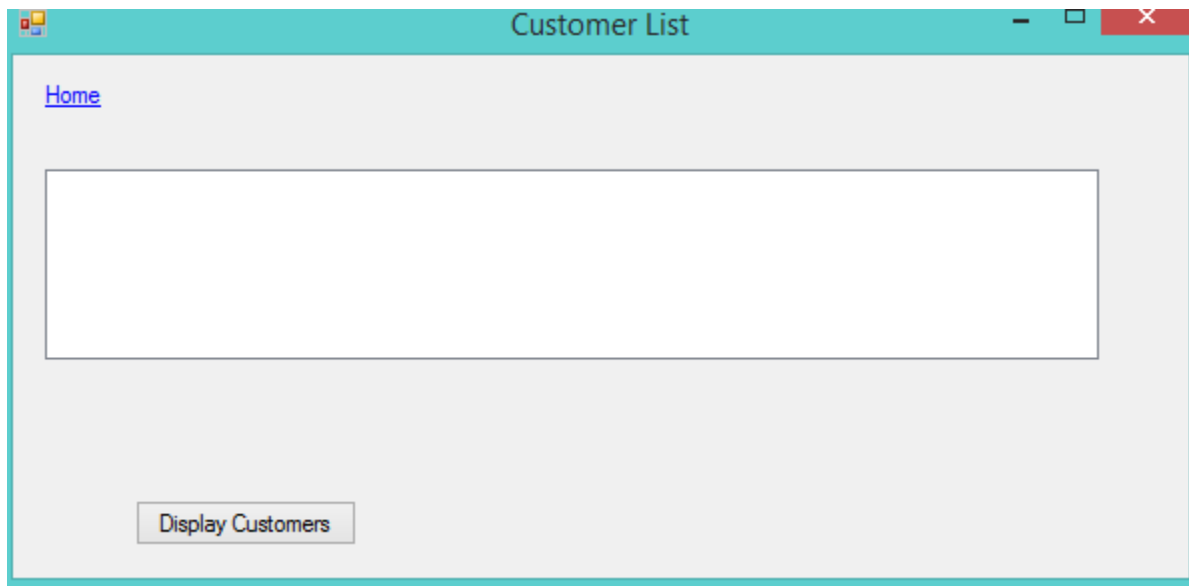
Customer Page

This is where a user would display the customers



Customer List Page

This is the page where the customers are displayed



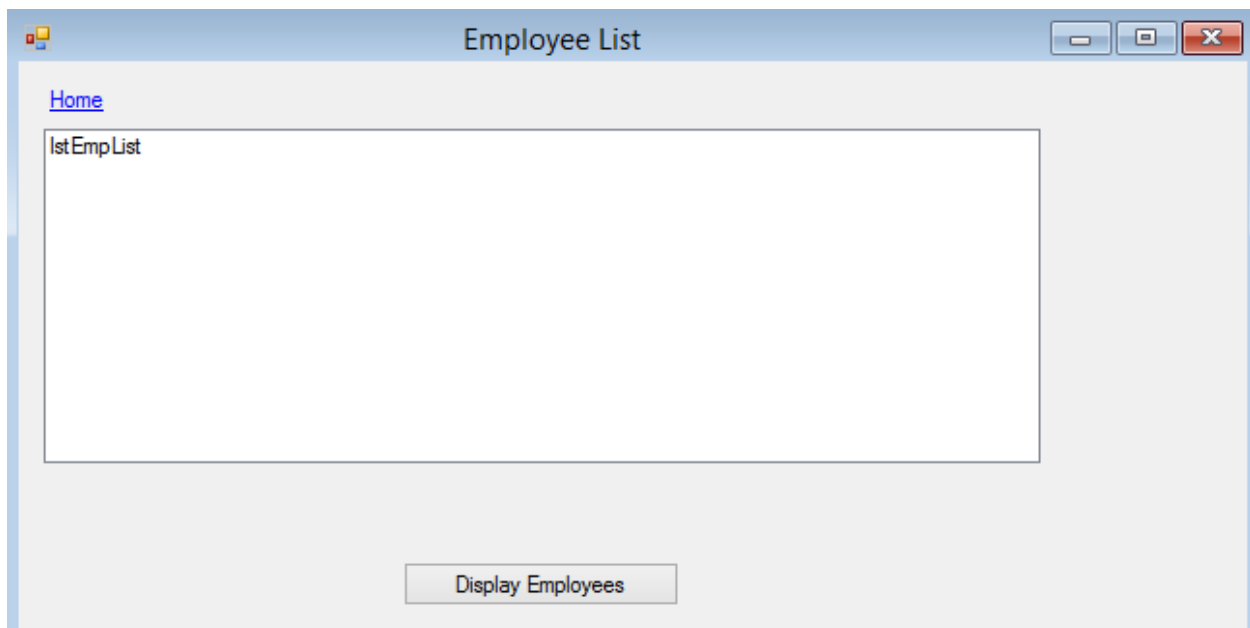
Employee Page

This is the page where the employee would go to see all the employees



Employee List Page

This is the page where the employees are displayed



Jobs Page

This is the page where a user would find the jobs to display



Jobs List Page

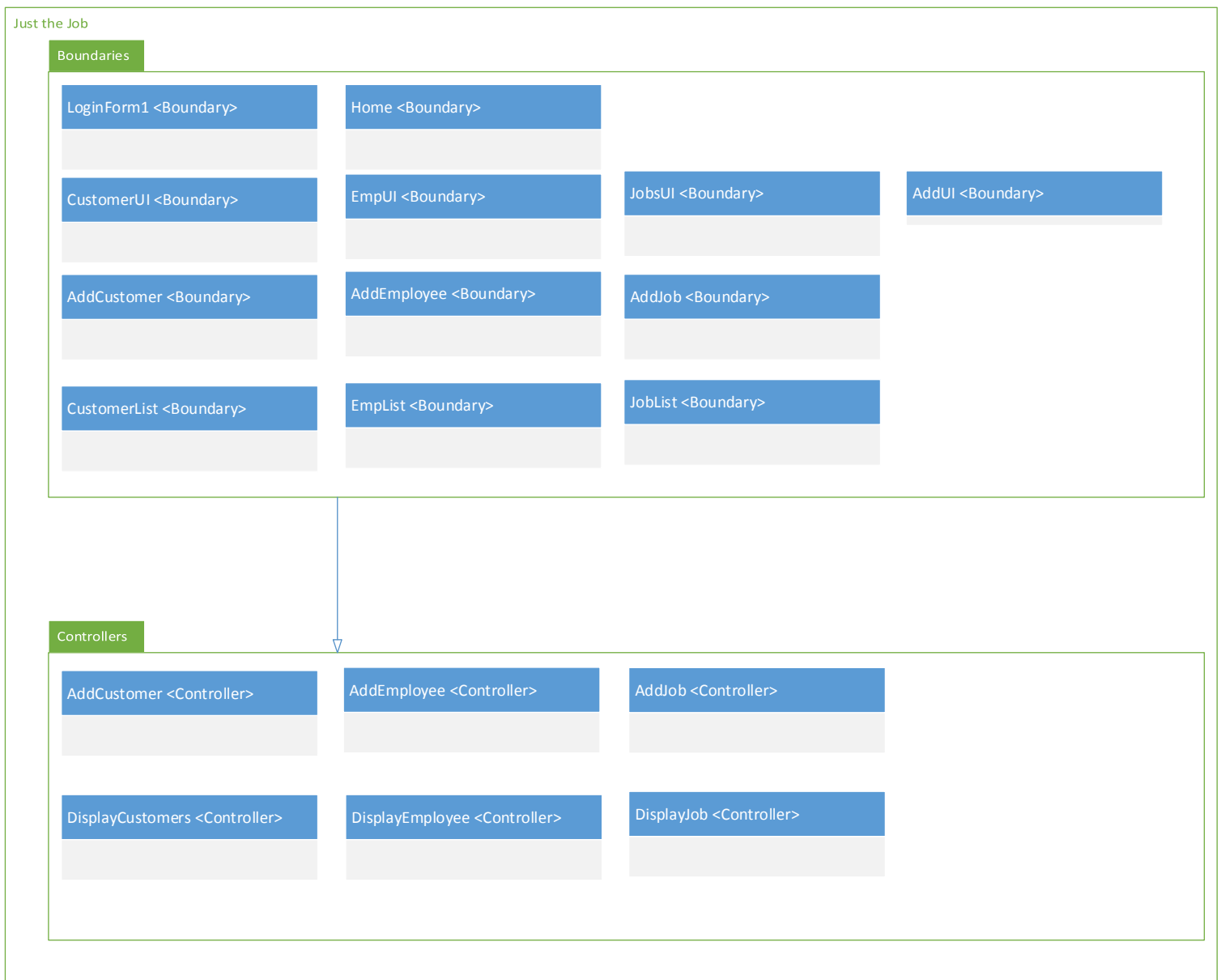
This is the page where the jobs are displayed



Package Diagrams for Design Organization

The package diagram below shows how our classes were organized (i.e. by class type). The top folder contains all of our boundary classes, which points to the second layer of controller classes.

This boundary to controller relationship was all that we implemented in Iteration 1, and the packaging reflects that. For a closer look at this diagram, please check Iteration 1 package diagram on our Google Drive



Implementation

Tested Code

We do not have our tested code from this iteration because we used the same Visual Basic file, overwriting the old code with our updated code. It is important to note that in this iteration, we did not implement entity classes. Entity classes were not needed because our lists held all of the information associated with each class. In general, our code was modeled in parallel with our design package diagram and design class diagram from this iteration, both of which were previously listed.

Package Diagrams for Code Organization

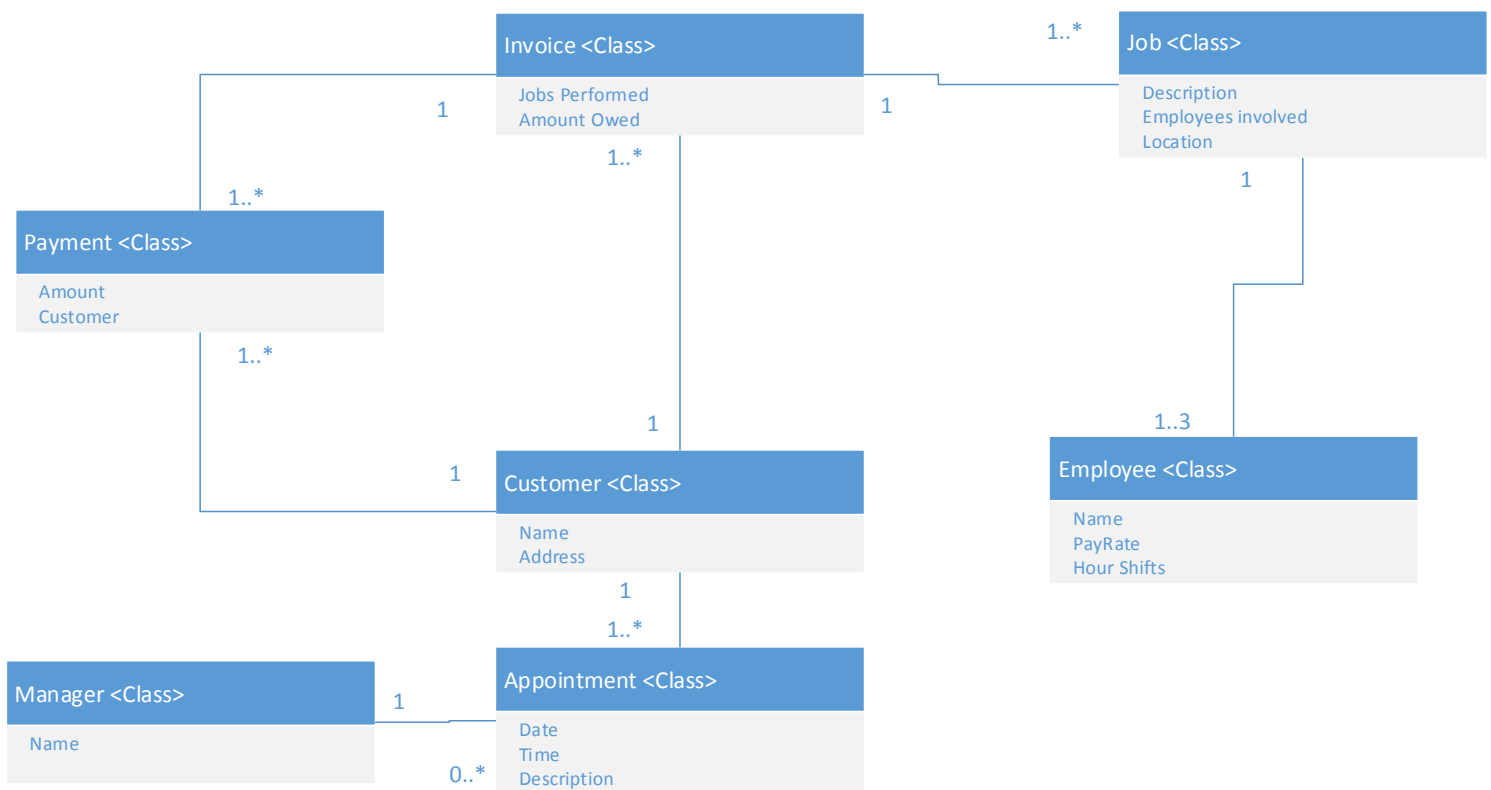
Our package diagram for Code Organization is identical to our design package diagram. See Iteration 1 design package diagram.

Chapter 2: Project Iteration 2

Analysis

Analysis Domain Model

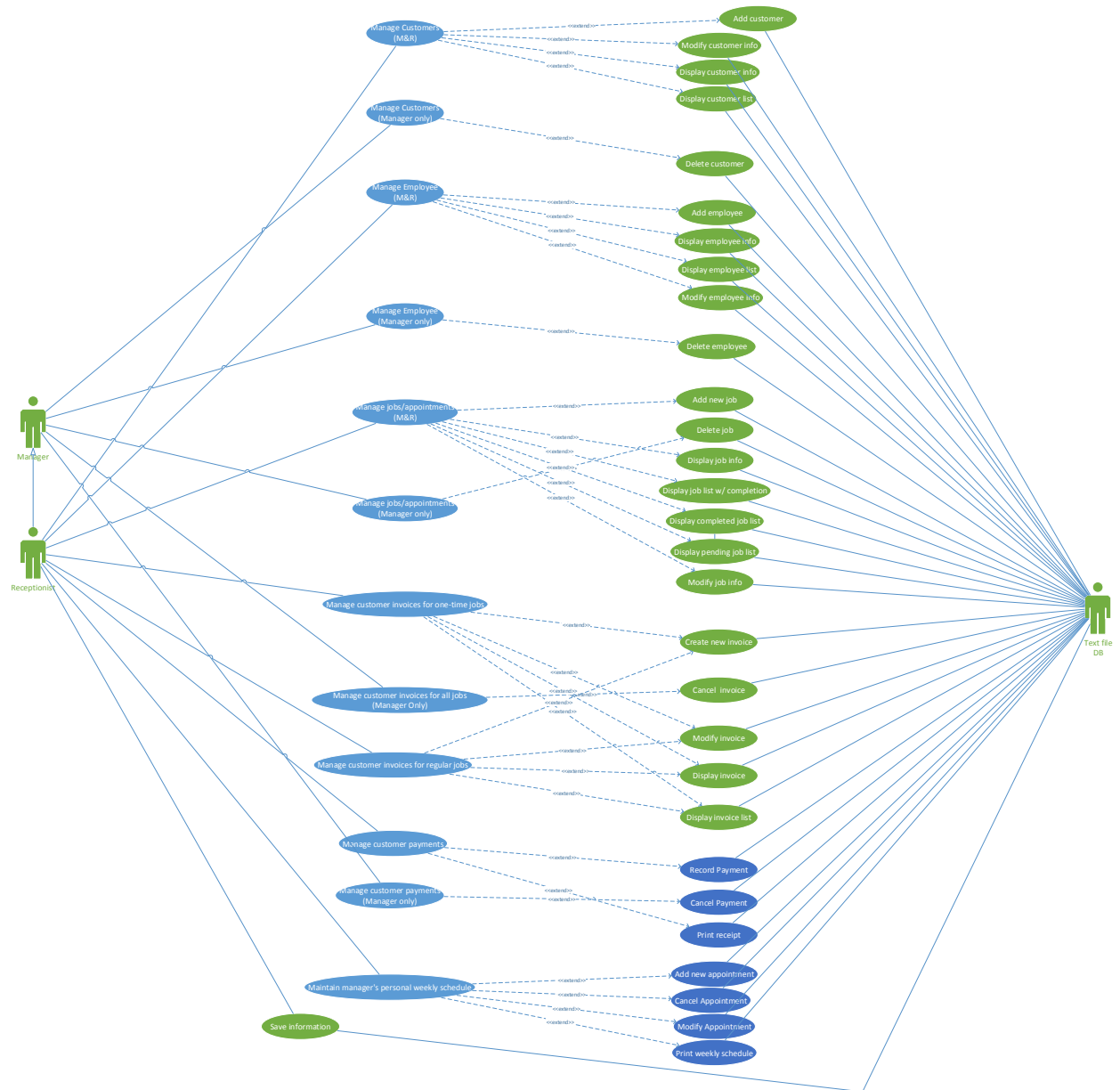
Our analysis domain model for Iteration 2 was very similar to our model from Iteration 1. There were not many changes needed because we still had a good plan for how our classes were going to interact. One modification we did make was an addition of a Manager class. Although we did not implement the Manager as a class in our final code, we still added the Manager class to the model for design purposes.



Requirements

Use-Case Diagram

Note: In iteration 2, we improved our use-case diagram greatly by adding the generalization for our two actors. This reduced the amount of lines and confusion of our diagram as the manager does everything the receptionist can do and more. Also, we updated our completed use cases to match our progress so far in the project and correlates directly with the software. The last update to this use-case diagram is the addition of the back end of writing to a text file. In our final use-case diagram, the only update is the completion of all requirements. To view the diagram at closer look, refer to our wiki page on Moodle.



Use Case Scenarios

Use Case: Delete existing Customer (1.b)

Actor(s): Manager

Goal: Remove an existing customer from the database

Overview: The manager should be able to remove the data about an existing customer if it is no longer needed.

Typical Course of Events:

Actor Action	System Response
1. The manager logs into the system	2. The system checks the credentials
3. They then go to the Customer section	
4. They then choose the delete customer link	
5. They choose which customer to delete	6. The chosen customer is removed

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Customer section; they will have to return to the home page.
4. They choose a different link on the Customer page, they will have to return to the customer.
5. They chose an invalid customer to remove, and are notified that they have done so and will then have to enter valid information.

Use Case: Modify information about an existing customer (1.c)

Actor(s): Receptionist, Manager

Goal: Edit the information stored about an existing customer

Overview: The manager/receptionist should be able to choose a customer, and then edit information about them.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. The System checks their credentials
3. They then go to the Customer page	
4. They then choose the modify customer option	
5. They choose which customer to modify, and change whatever information they want	6. These changes are saved to the database.

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Customer section; they will have to return to the home page.
4. They choose a different link on the Customer page, they will have to return to the customer.
5. They choose an invalid customer to modify, they will be notified. They will then have to choose a correct customer.

Use Case: Display/print information about an existing customer based on his/her id or name (1.d)

Actor(s): Receptionist, Manager

Goal: Display /print information about an existing customer using their id or name

Overview: After choosing a customer by entering their name or id, all of that specific customer's information should be displayed.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated.
3. They go to the customer page	
4. They then go to the display page.	
5. They choose the display individual option, and enter a name or id.	6. The system displays the correct customer.

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Customer section; they will have to return to the home page.
4. They choose a different link on the Customer page, they will have to return to the customer page.
5. They enter an invalid name or id, they will be prompted that they have done so. They should then enter the correct information.

Use Case: Delete an existing employee (2.b)

Actor(s): Manager

Goal: Remove an existing employee from the database

Overview: The manager has the option to delete an employee from the database.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated.
3. They go to the employee page	
4. They choose the delete employee option	
5. They then choose the employee to delete	6. The system then removes that employee from the database

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Employee section; they will have to return to the home page.
4. They choose a different link on the Employee page, they will have to return to the employee page.
5. They choose an invalid employee, they are prompted that they have done so. They then can enter the correct information.

Use Case: Display print information about an existing employee (2.c)

Actor(s): Receptionist, Manager

Goal: Display the information about an existing employee

Overview: The manager/receptionist choose a specific employee, and the system displays their information.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated
3. They go to the employee page	
4. They choose the Display employees link	
5. They choose the display individual option, and indicate which employee they want	6. Their information is displayed

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Employee section; they will have to return to the home page.
4. They choose a different link on the Employee page, they will have to return to the employee page.
5. They choose an invalid employee, they are prompted that they have done so. They then can enter the correct information.

Use Case: Delete/Cancel Job(3.b)

Actor(s): Manager

Goal: Delete a job form the database that has been cancelled.

Overview: The manager can choose a job to remove form the database.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated
3. They go to the Job page	
4. They choose the delete job link	
5. They choose a job they would like to be deleted	6. The job is removed from the database

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Job section; they will have to return to the home page.
4. They choose a different link on the Job page, they will have to return to the Job page.
5. They choose an invalid job, they are prompted that they have done so. They then can enter the correct information.

Use Case: Print the status of a job (3.c)

Actor(s): Receptionist, Manager

Goal: Display the status of job

Overview: The Receptionist/Manager should be able to choose a job and see its completion status.

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated
3. They go to the Jobs page	
4. They go to the Display Jobs page	
5. They choose the individual option, and then choose a job to display.	6. The information is displayed.

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt
3. They go to a section different than the Job section; they will have to return to the home page.
4. They choose a different link on the Job page, they will have to return to the Job page.
5. They choose an invalid job, they are prompted that they have done so. They then can enter the correct information.

Use Case: Display/print a list of all completed jobs (3.e)

Actor(s): Receptionist, Manager

Goal: To display the list of completed jobs in the system

Overview: The receptionist/manager should be able to enter the system and display the list of all completed jobs

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated
3. They go to the Jobs page	
4. They go to the Display Jobs page	
5. They choose to display jobs by completion	6. The completed jobs are displayed

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Job section; they will have to return to the home page.
4. They choose a different link on the Job page, they will have to return to the Job page.

Use Case: Display/print a list of all pending jobs (3.f)

Actor(s): Receptionist, Manager

Goal: To display a list of all pending jobs in the system

Overview: The receptionist/manager should be able to enter the system and display a list of all pending jobs

Typical Course of Events:

Actor Action	System Response
1. They log in	2. Their credentials are validated
3. They go to the Jobs page	
4. They go to the Display Jobs page	
5. They choose to display jobs by pending status	6. The pending jobs are displayed

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Job section; they will have to return to the home page.
4. They choose a different link on the Job page, they will have to return to the Job page.

Use Case: Create a new invoice (4.a)

Actor(s): Receptionist, Manager

Goal: To create a new invoice and put it in the database

Overview: The receptionist/manager should be able to access the system and create a new invoice.

Typical Course of Events:

Actor Action

1. They log in
3. They go to the Invoice page
4. They choose to create new invoice
5. All of the invoice information is entered
6. The invoice is saved

System Response

2. Their credentials are validated
7. The invoice is entered into the system

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Invoice section; they will have to return to the home page.

Use Case: Cancel an existing invoice (4.b)

Actor(s): Manager

Goal: To cancel an existing invoice

Overview: The manager should be able to access the system and cancel an existing invoice in the system.

Typical Course of Events:

Actor Action

1. They log in
3. They go to the Invoice page
4. They search for a specific invoice by ID
5. They choose to cancel the invoice

System Response

2. Their credentials are validated
6. The invoice is cancelled

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Invoice section; they will have to return to the home page.
4. The wrong Invoice ID is entered. They can always search again.

Use Case: Modify an existing invoice (4.c)

Actor(s): Receptionist, Manager

Goal: To modify the info in a current invoice

Overview: The receptionist/manager should be able to access the system and modify the information in an existing invoice.

Typical Course of Events:

Actor Action

1. They log in
3. They go to the Invoice page
4. They search for an invoice by ID
5. They choose to edit the invoice
6. They enter the new info and save

System Response

2. Their credentials are validated
7. The invoice is re-entered into the system

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Invoice section; they will have to return to the home page.
4. The wrong Invoice ID is entered. They can always search again.

Use Case: Display/print an existing invoice (4.d)

Actor(s): Receptionist, Manager

Goal: To display an individual existing invoice

Overview: The receptionist/manager should be able to access the system and display or print an existing invoice.

Typical Course of Events:

Actor Action

1. They log in
3. They go to the Invoice page
4. They search for an Invoice by ID
5. They choose to print/display Invoice

System Response

2. Their credentials are validated
6. Invoice is displayed

Alternative courses:

- 1-2. They provide incorrect information and will have to reattempt.
3. They go to a section different than the Invoice section; they will have to return to the home page.
4. The wrong Invoice ID is entered. They can always search again.

Use Case: Display/print a list of all invoices and their payment status (4.e)

Actor(s): Receptionist, Manager

Goal: To display a list of all invoices, along with their payment status

Overview: The receptionist/manager should be able to access the system and display a full list of invoices and their payment status

Typical Course of Events:

Actor Action

System Response

1. They log in

2. Their credentials are validated

3. They go to the Invoice page

4. They choose to display all invoices

5. All invoices are displayed

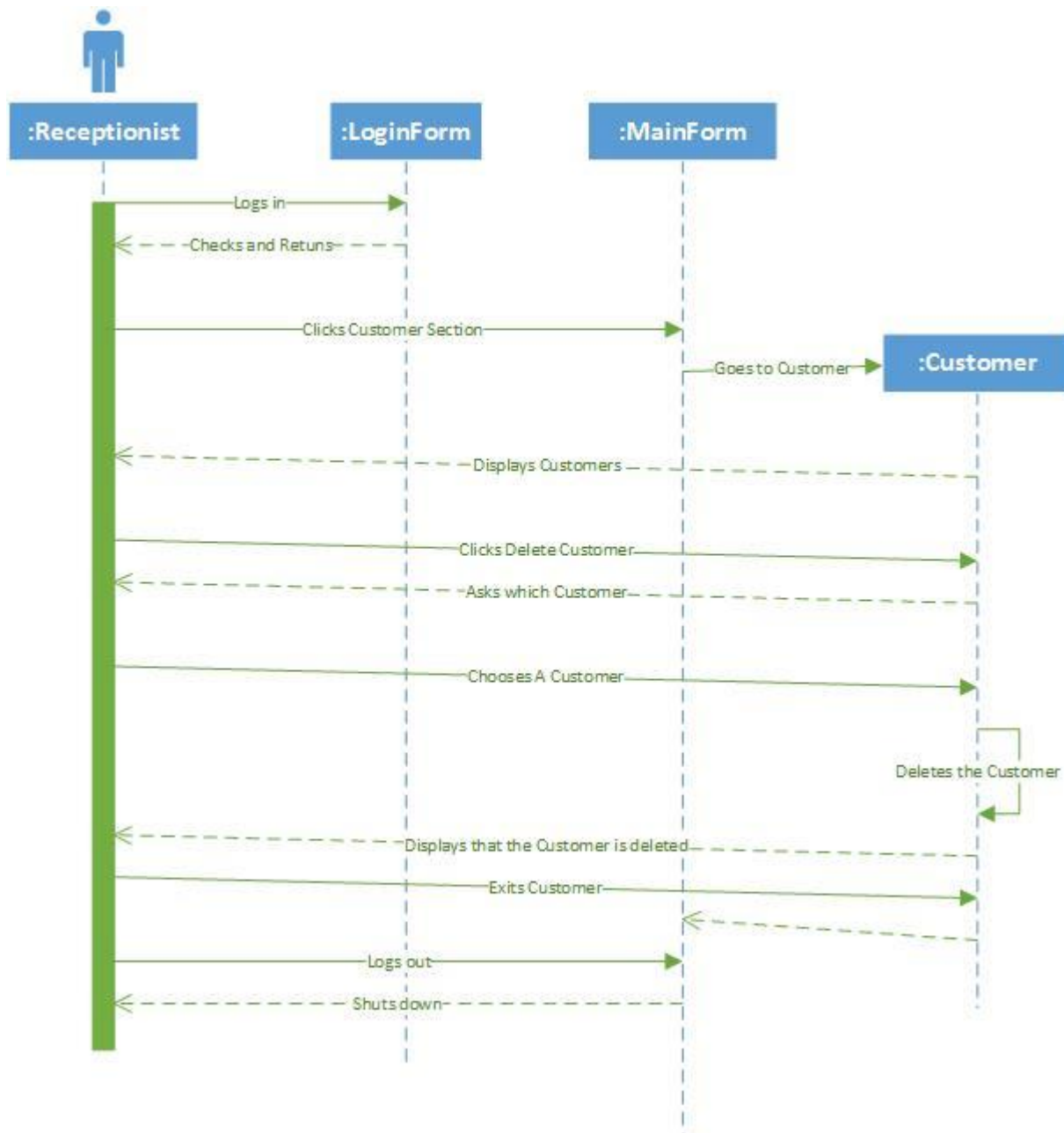
Alternative courses:

1-2. They provide incorrect information and will have to reattempt.

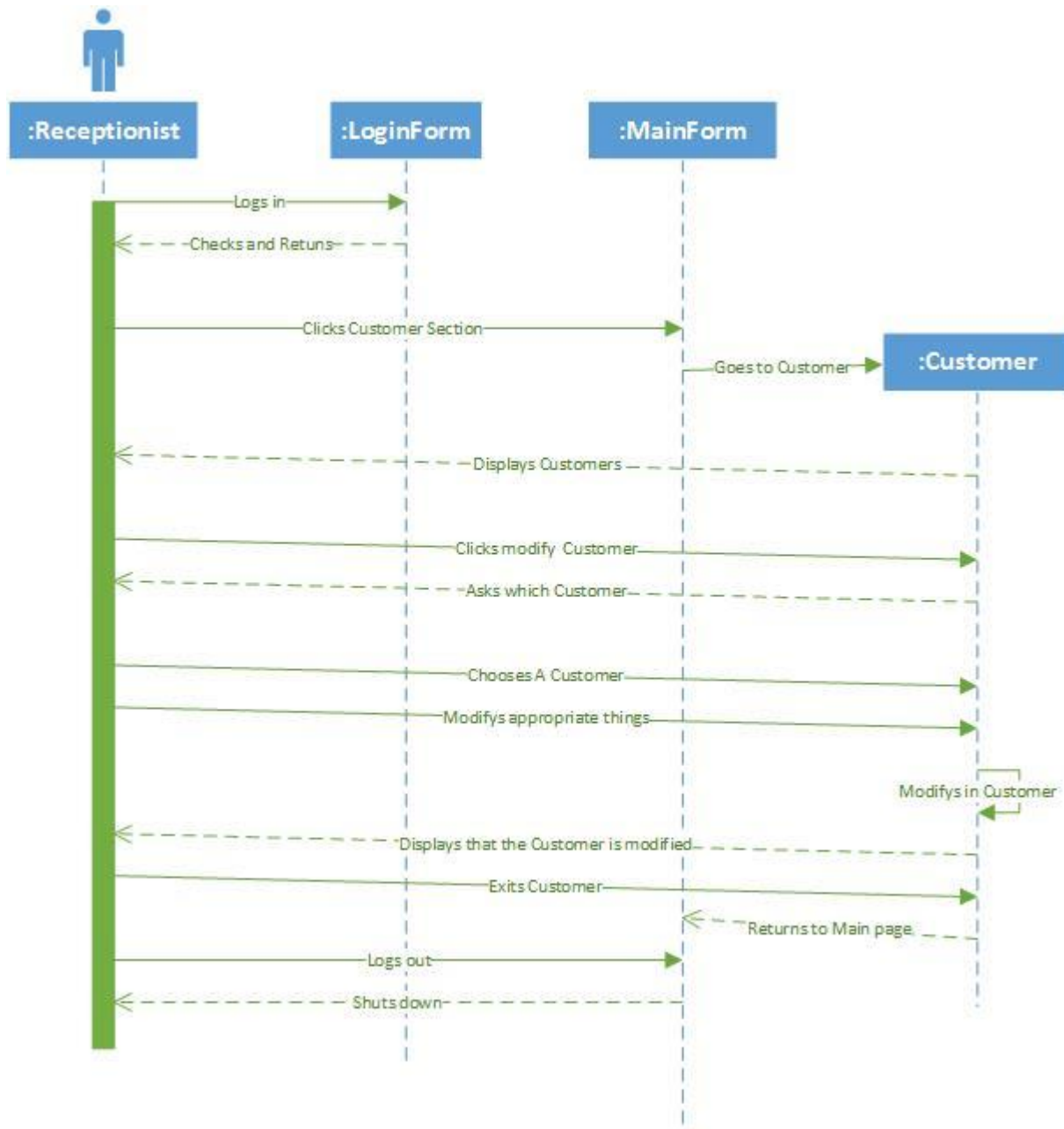
3. They go to a section different than the Invoice section; they will have to return to the home page.

High-Level Sequence Diagram

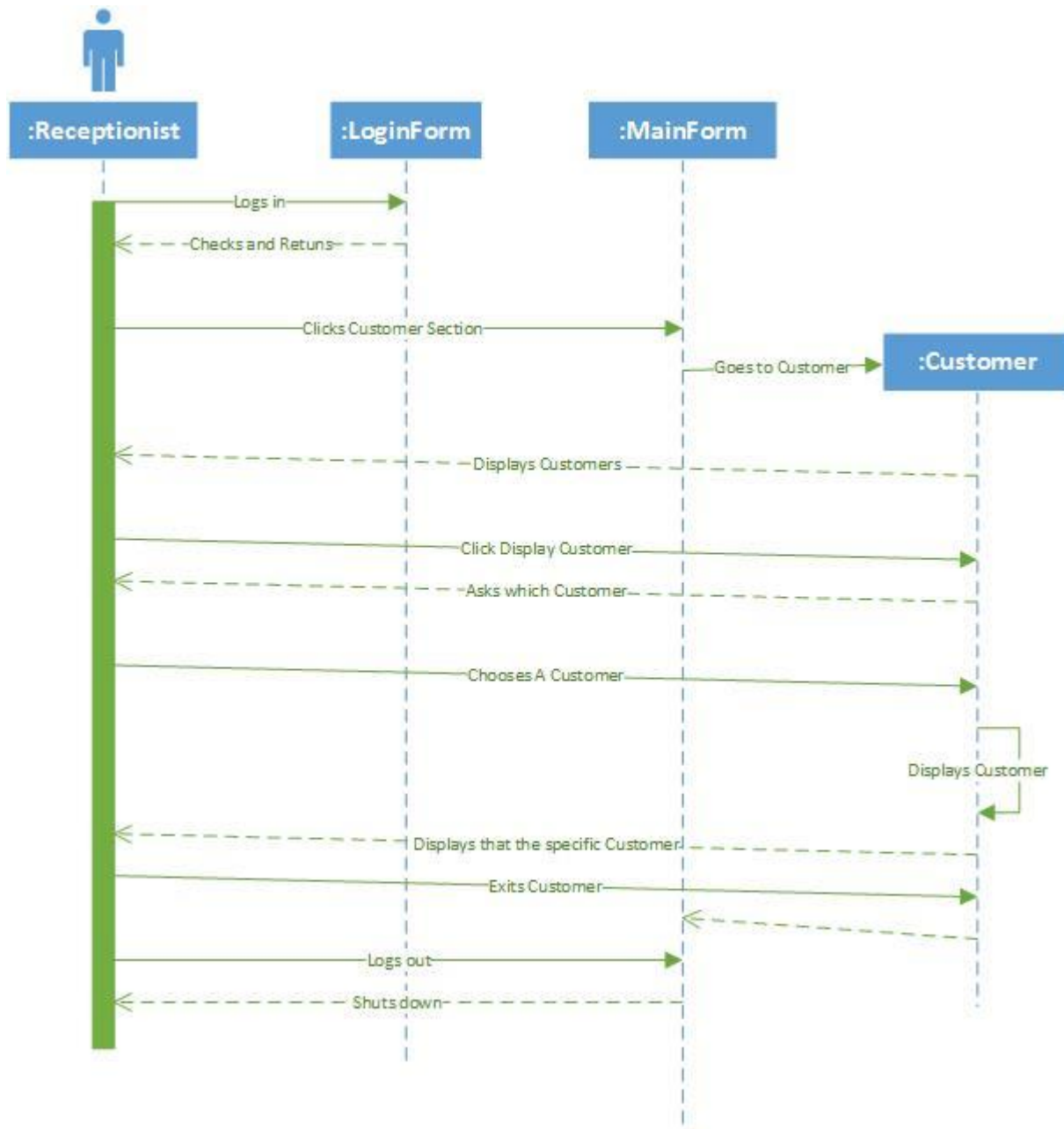
This is the High-level Sequence Diagram for Requirement 1b) Delete an existing customer



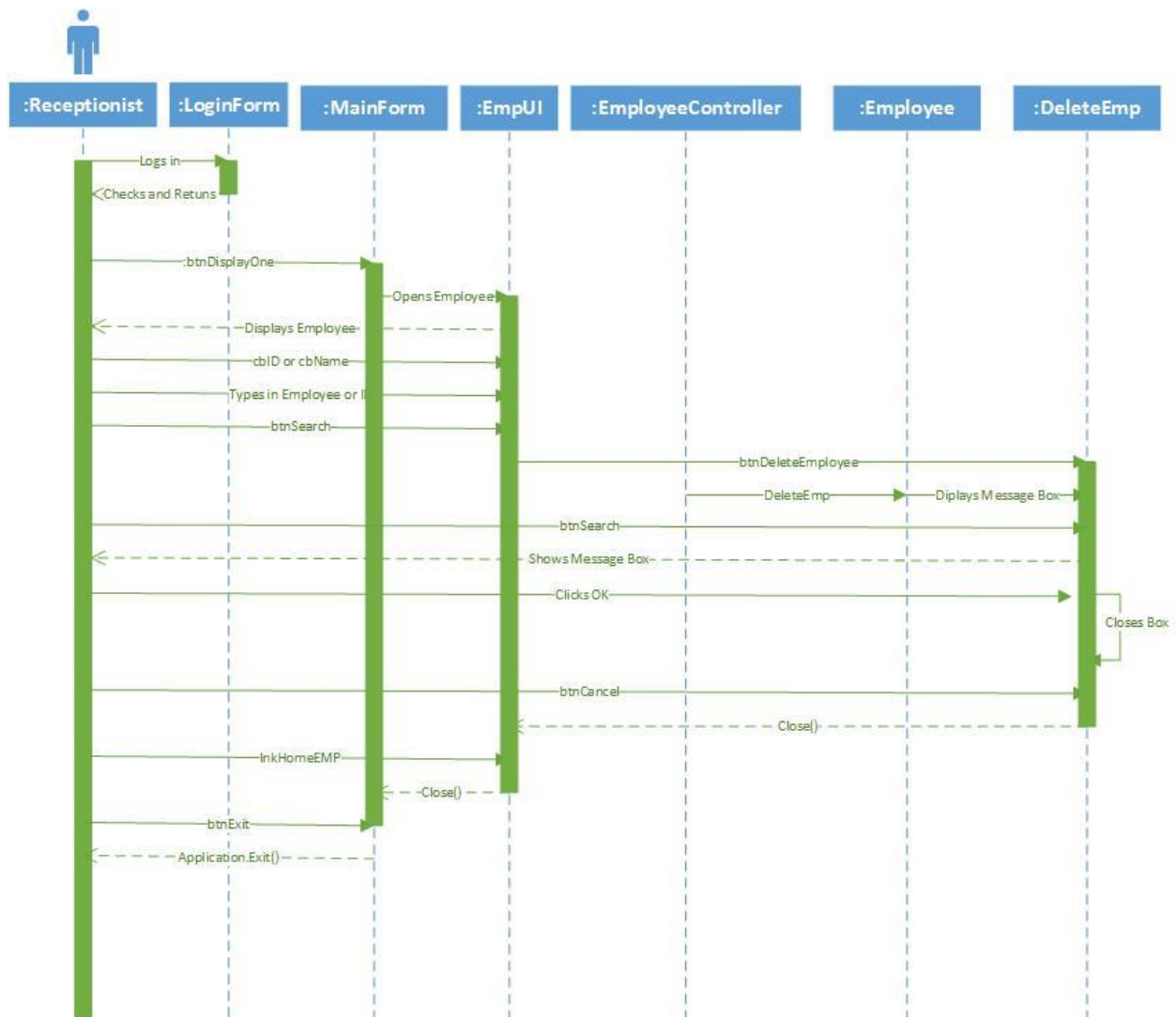
This is the High-level Sequence Diagram for Requirement 1c) Modify information stored about an existing customer



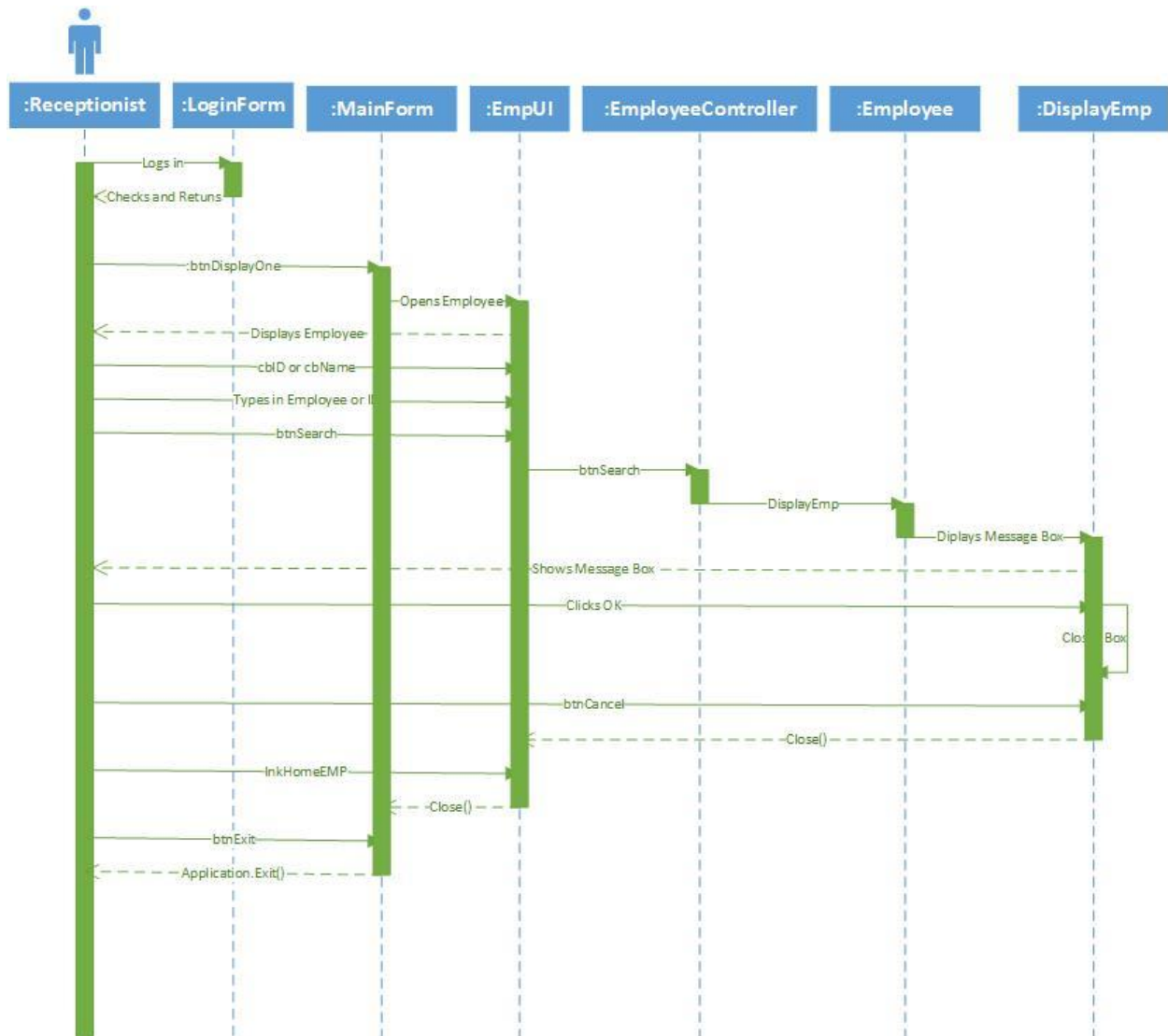
This is the High-level Sequence Diagram for 1d) Display/print information about an existing customer based on his/her id or name



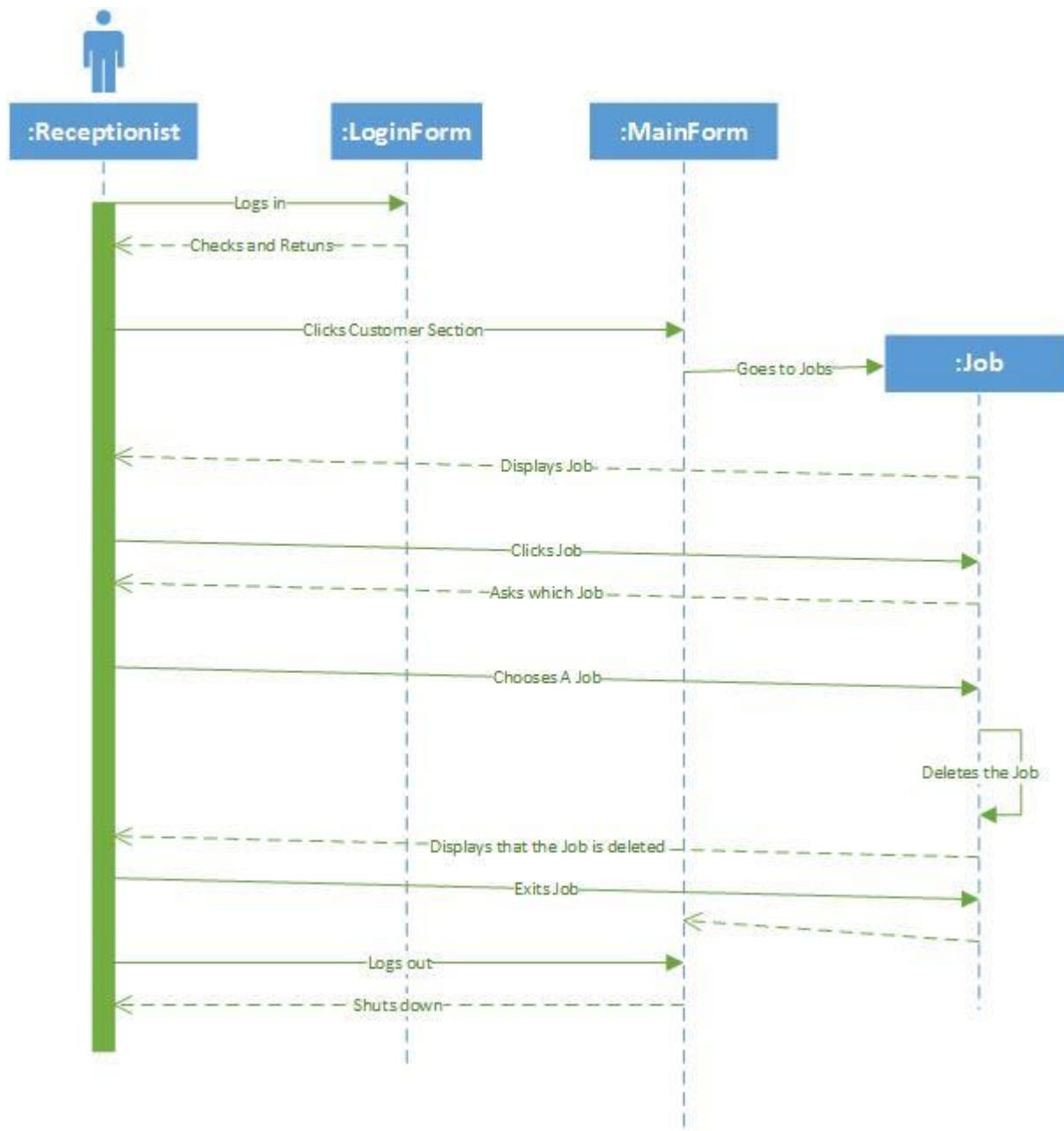
This is the High-level Sequence Diagram for 2b) Delete an existing employee



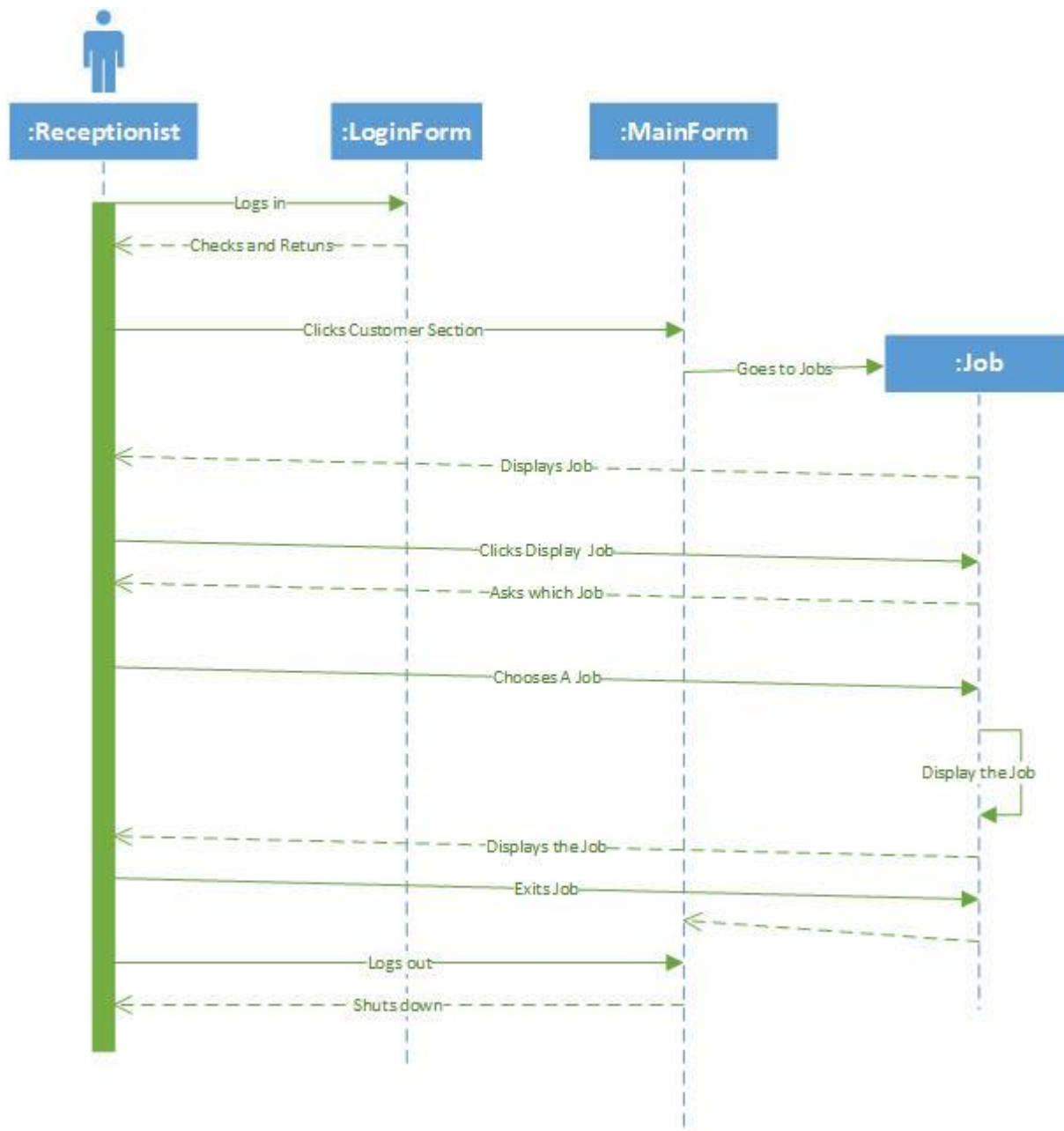
This is the High-level Sequence Diagram for 2c) Display/print information about an existing employee



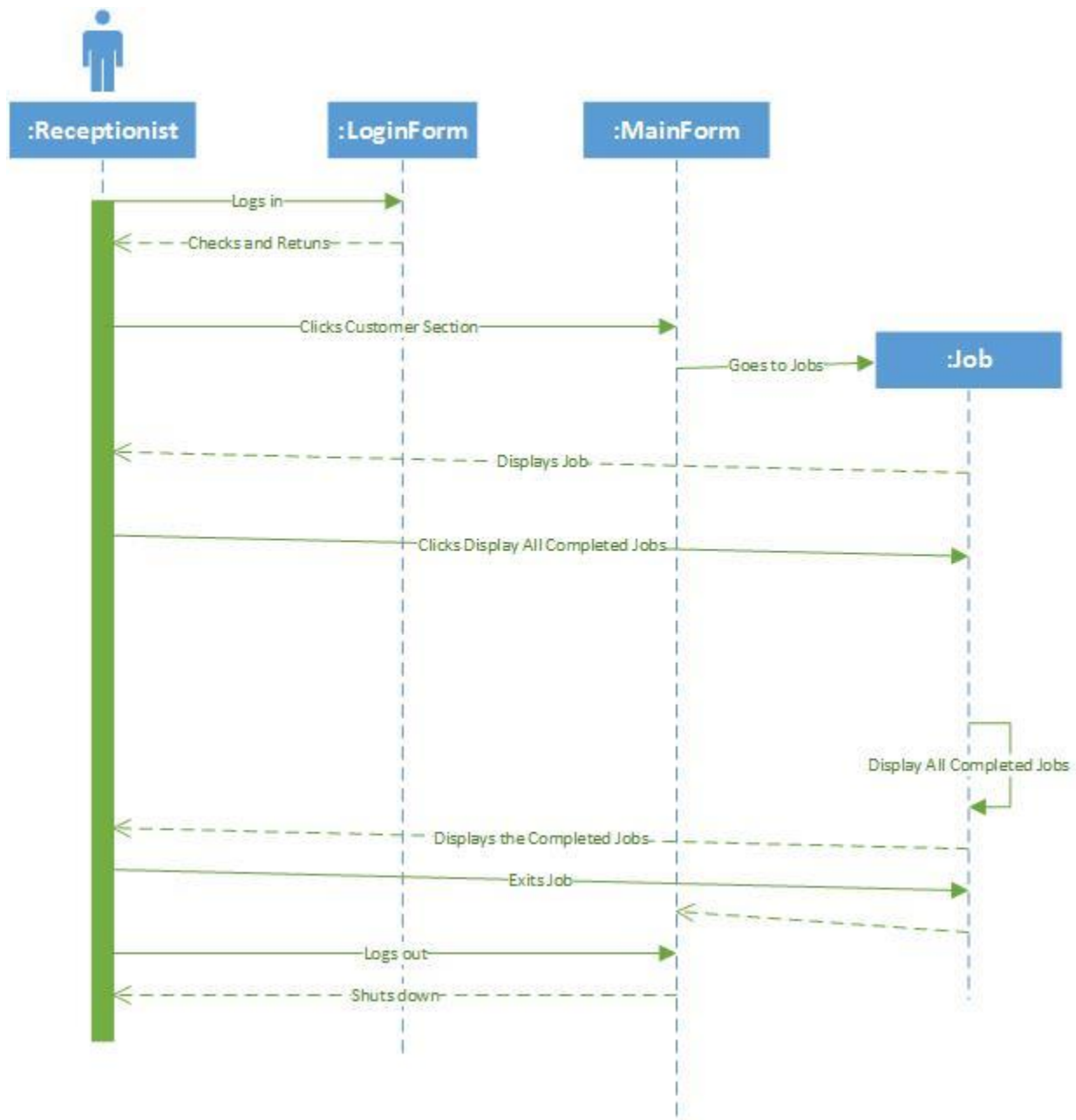
This is the High-level Sequence Diagram for 3b) Delete/cancel an existing job



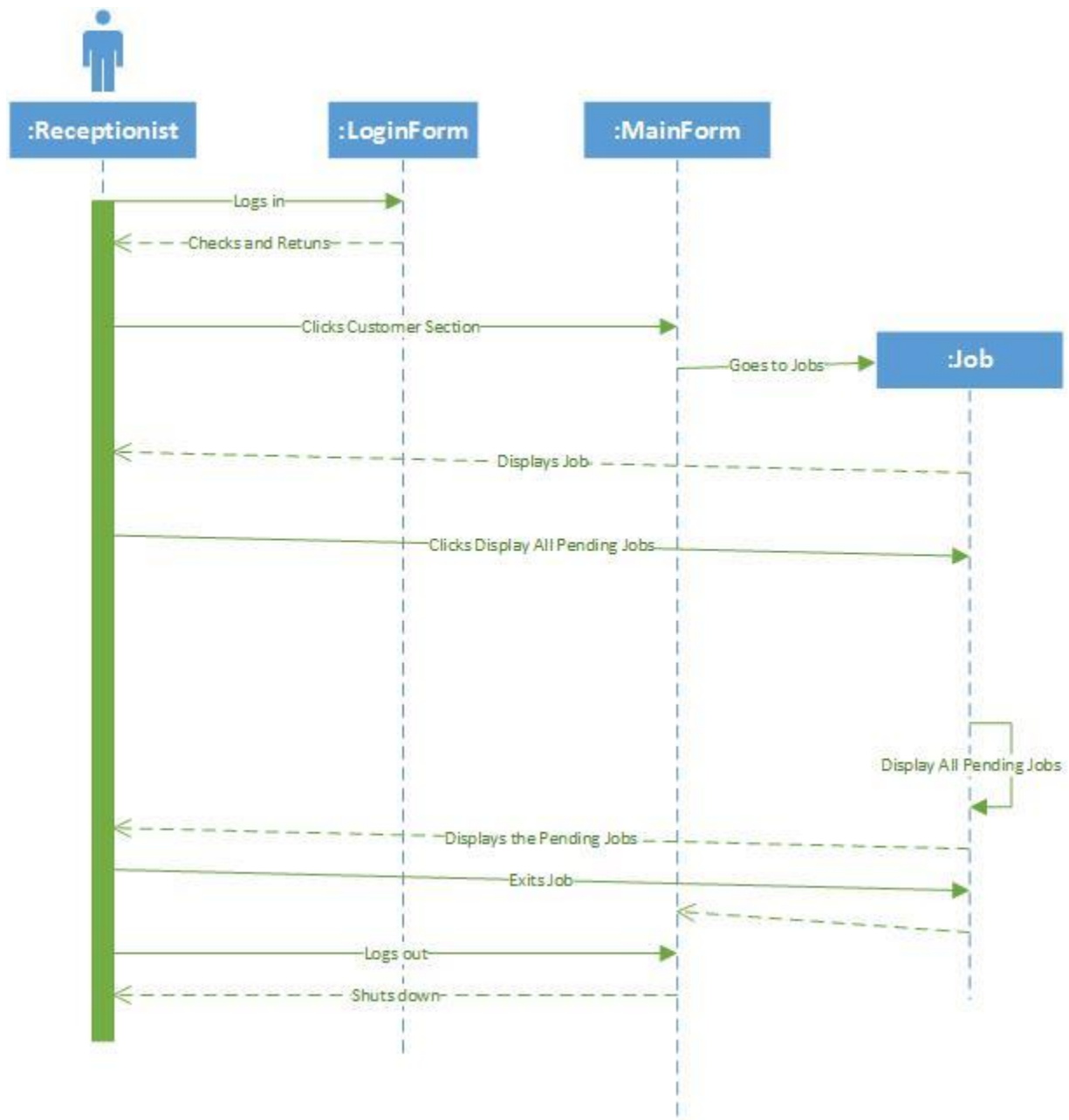
This is the High-level Sequence Diagram for 3c) Display/print the status of a specific job



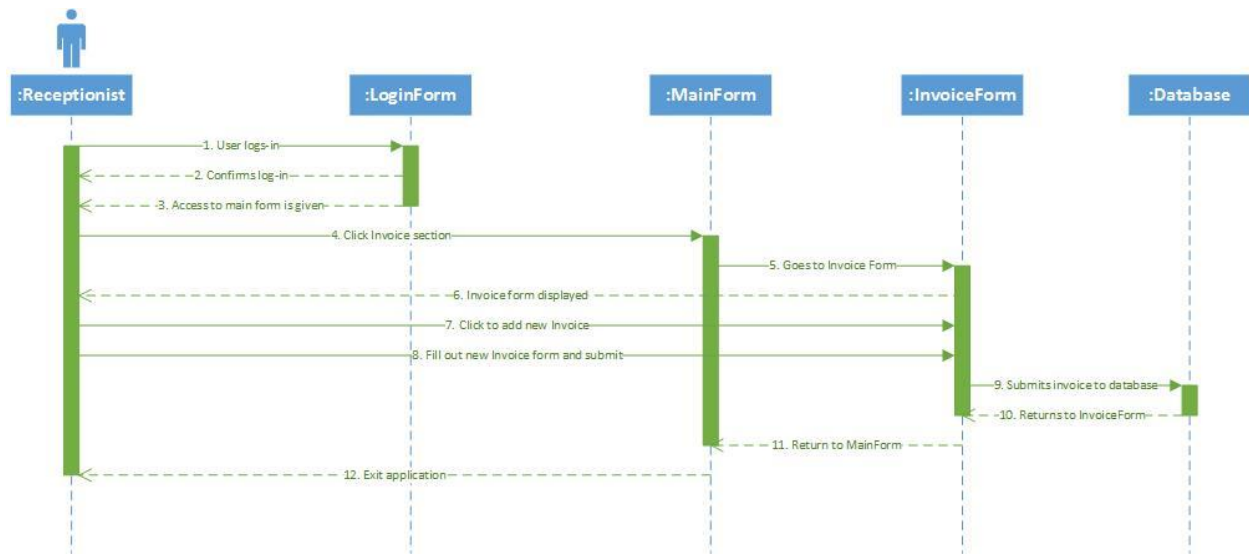
This is the High-level Sequence Diagram for 3e) Display/print a list of all completed jobs



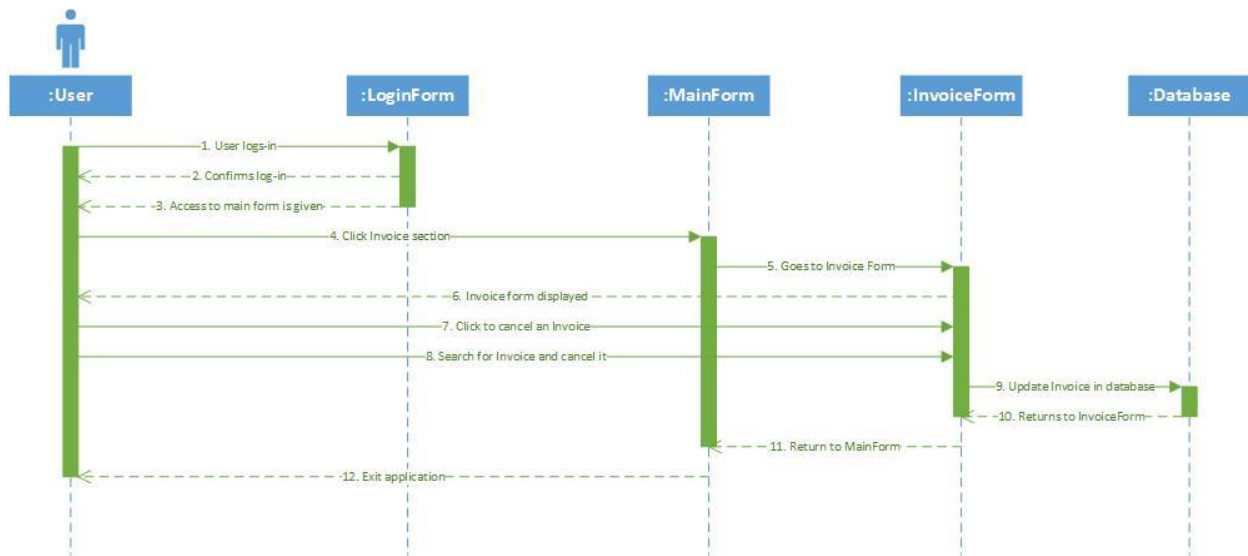
This is the High-level Sequence Diagram for 3f) Display/print a list of all pending jobs



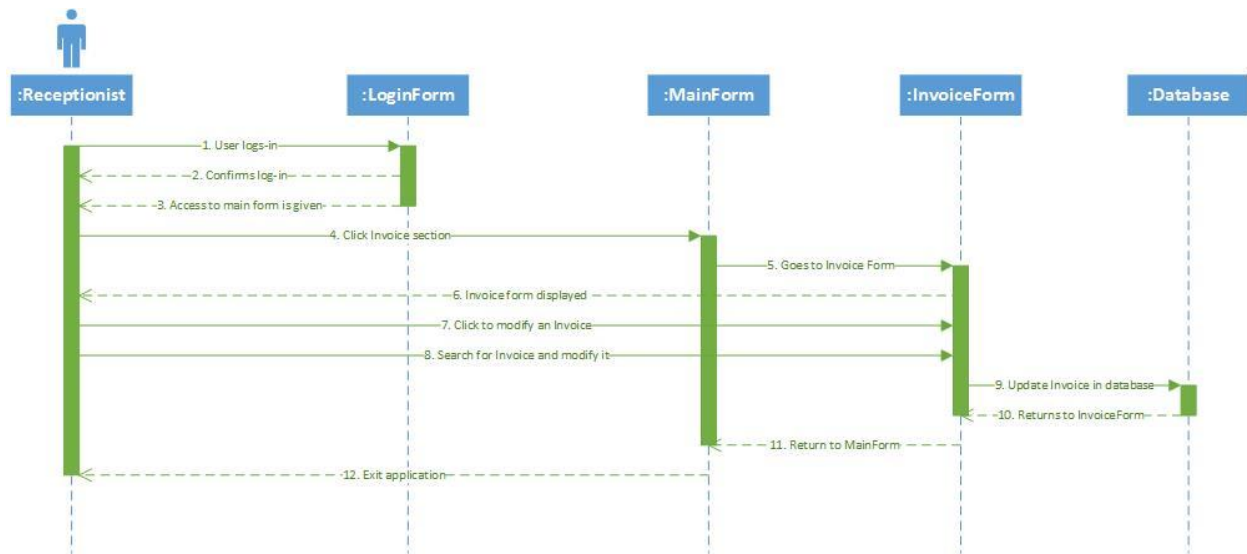
This is the High-level Sequence Diagram for 4a) Create a new invoice



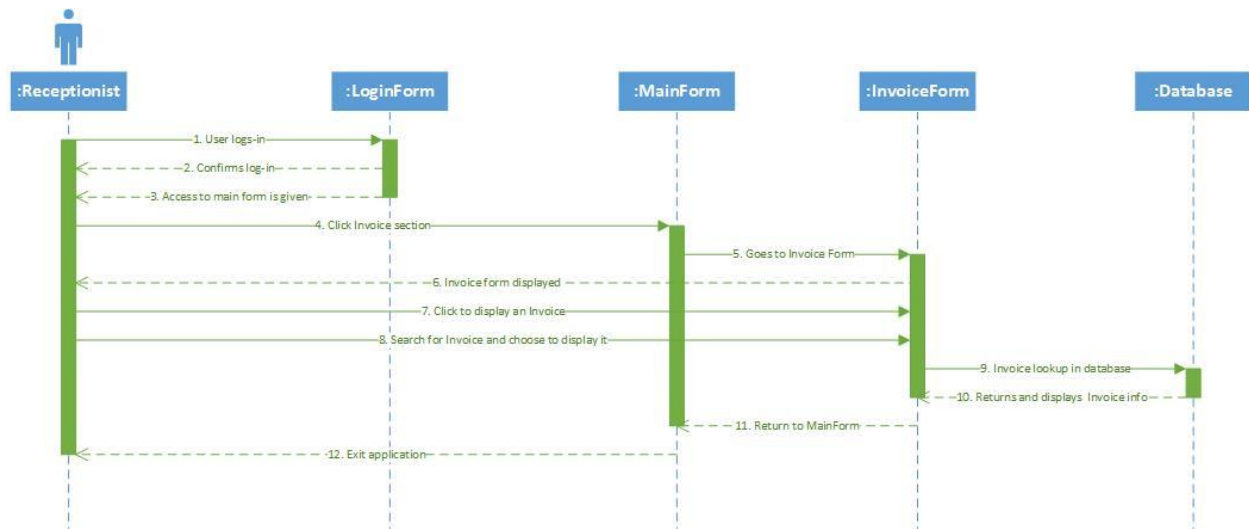
This is the High-level Sequence Diagram for 4b) Cancel an existing invoice



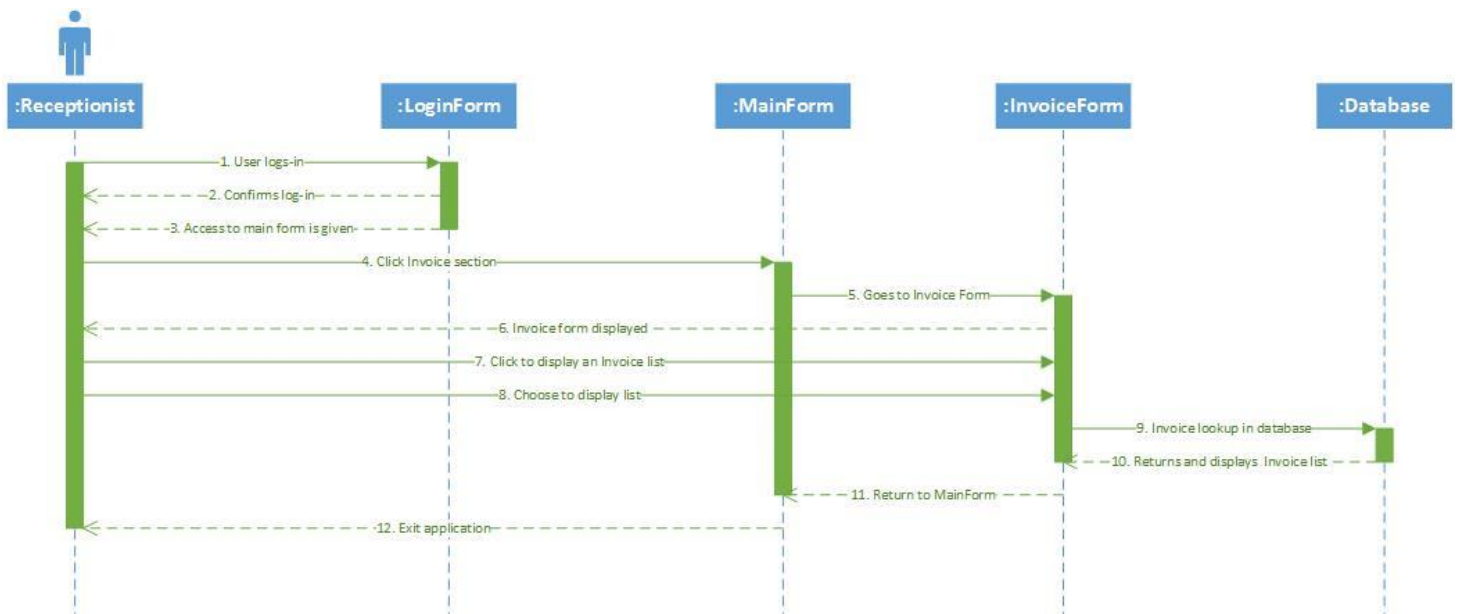
This is the High-level Sequence Diagram for 4c) Modify an existing invoice



This is the High-level Sequence Diagram for 4d) Display/print an existing invoice



This is the High-level Sequence Diagram for 4e) Display/print a list of all invoices and their payment status



CRC Cards

Note:

In this iteration, we developed our back end of our software by including our entities, so that we could write objects to the text files and add objects to list boxes. Also, we redesigned our controller classes to more generic controllers with more functions for a single entity.

Customer <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide Customer Info (Name, Phone, Address, Repeat, ID)	

Job <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Store job info (When, Where, ID, Employees Assigned)	Employee

Invoice <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Store invoice info like customer ID, Job ID, and whether it is paid or not	Customer, Job

Employee <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide information about employee	

CustomerController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
Add Customer	Customer
Load Customer	Customer
Check ID if exists	Customer
Modify Customer	Customer
Delete Customer	Customer
Display Customer	Customer

EmployeeController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
Add Employee	Customer, Employee
Load Employee	Employee
Check ID if exists	Customer, Employee
Modify Employee	Customer, Employee
Delete Employee	Employee
Display Employee	Employee

InvoiceController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Invoice -Load Invoice -Edit Invoices when modifying or adding Jobs -Check to see if IDs exist and get IDs -Apply Payments to Invoices -Delete Invoice -Display Invoices As All or One -Create Invoice ID	Customer, Job, Invoice Job Job, Invoice Customer, Job, Invoice Customer, Invoice, Job Invoice Invoice Invoice

JobController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Job -Load Jobs -Check to see if ID exists for customer and employee -Modify Job -Delete Job -Display All Jobs, a Job, or jobs in a week -check to see what invoices are worked on for the week	Customer, Employee, Job, Invoice Job Customer, Employee Customer, Employee, Job, Invoice Job Job Job, Invoice

High-Level Analysis Class Diagram

At this point in Iteration 2, we did not develop a full analysis class diagram that included all necessary boundaries, controllers, and entities. However, our detailed class diagram (shown on the next page) for this iteration does show all necessary boundaries, controllers, and entities up to this point.

Package Diagrams for Requirements

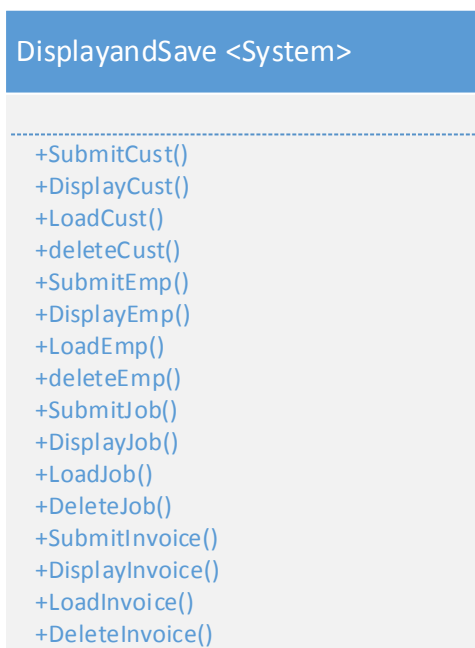
Because we did not have a complete analysis class diagram at this point in Iteration 1, we could not organize all of our classes in package form. For a package diagram covering all of the requirements from Iteration 2, please see our Iteration 2 design package diagram.

Design

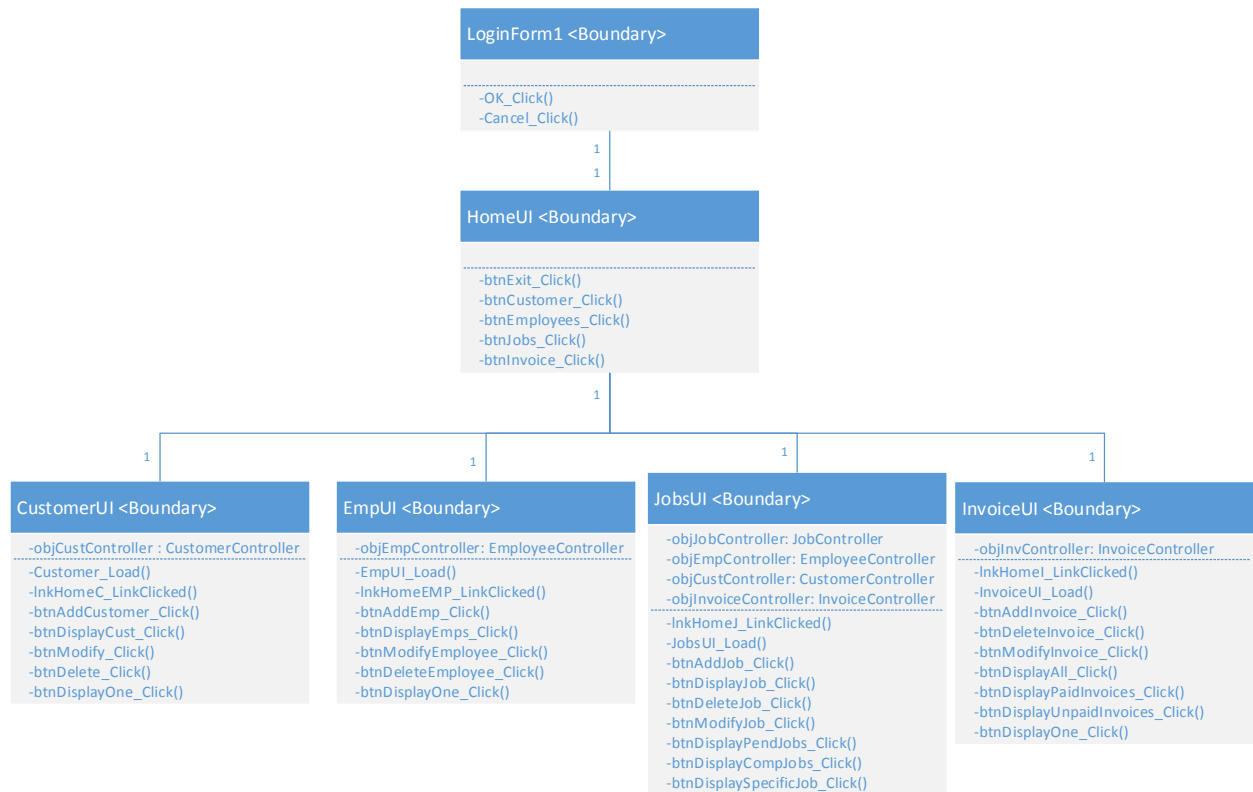
Detailed Design Class Diagram

Our detailed class diagram for Iteration 2 is broken up over the next 5 pages. For a complete look at our full class diagram, please refer to our Wiki. The first class posted is our Display and Save system class. This class is instantiated throughout our controllers, as information is displayed and saved automatically in a majority of our List Boxes. Our outer boundaries are the first few forms that the user encounters, and much GUI design went into those forms. From there, the diagram is broken down by entity class, showing all three layers of the architecture.

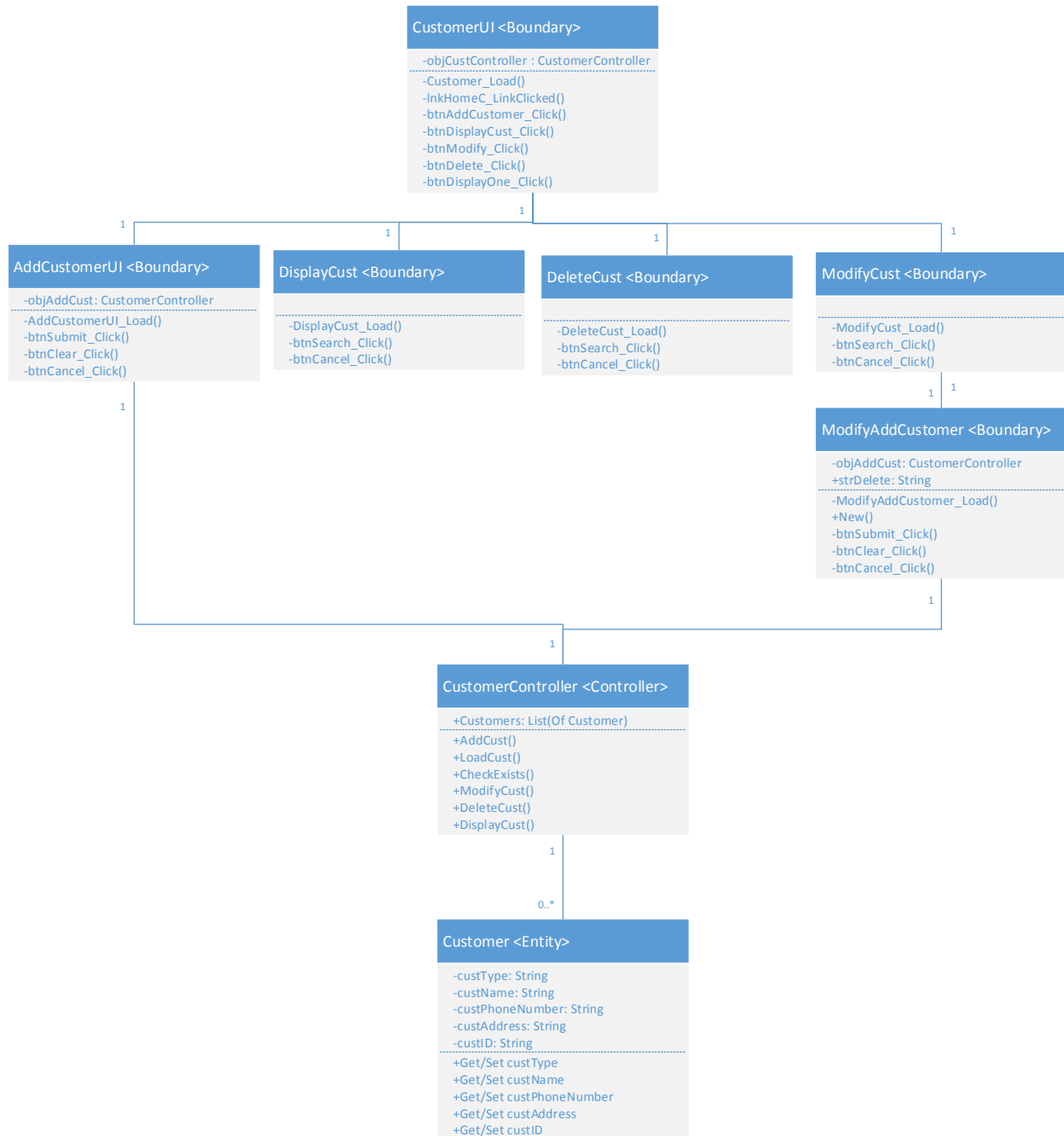
System Class



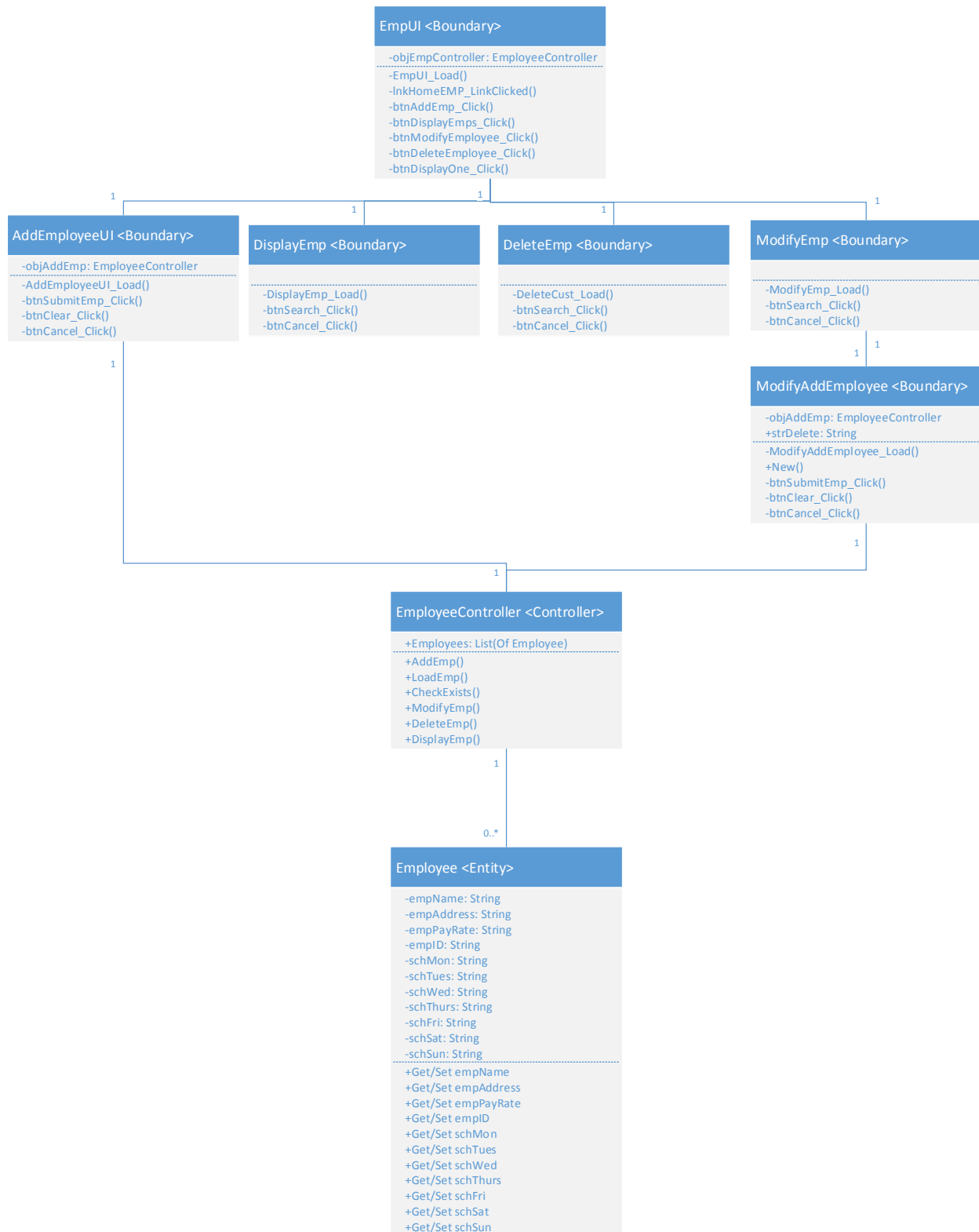
Outer Boundaries



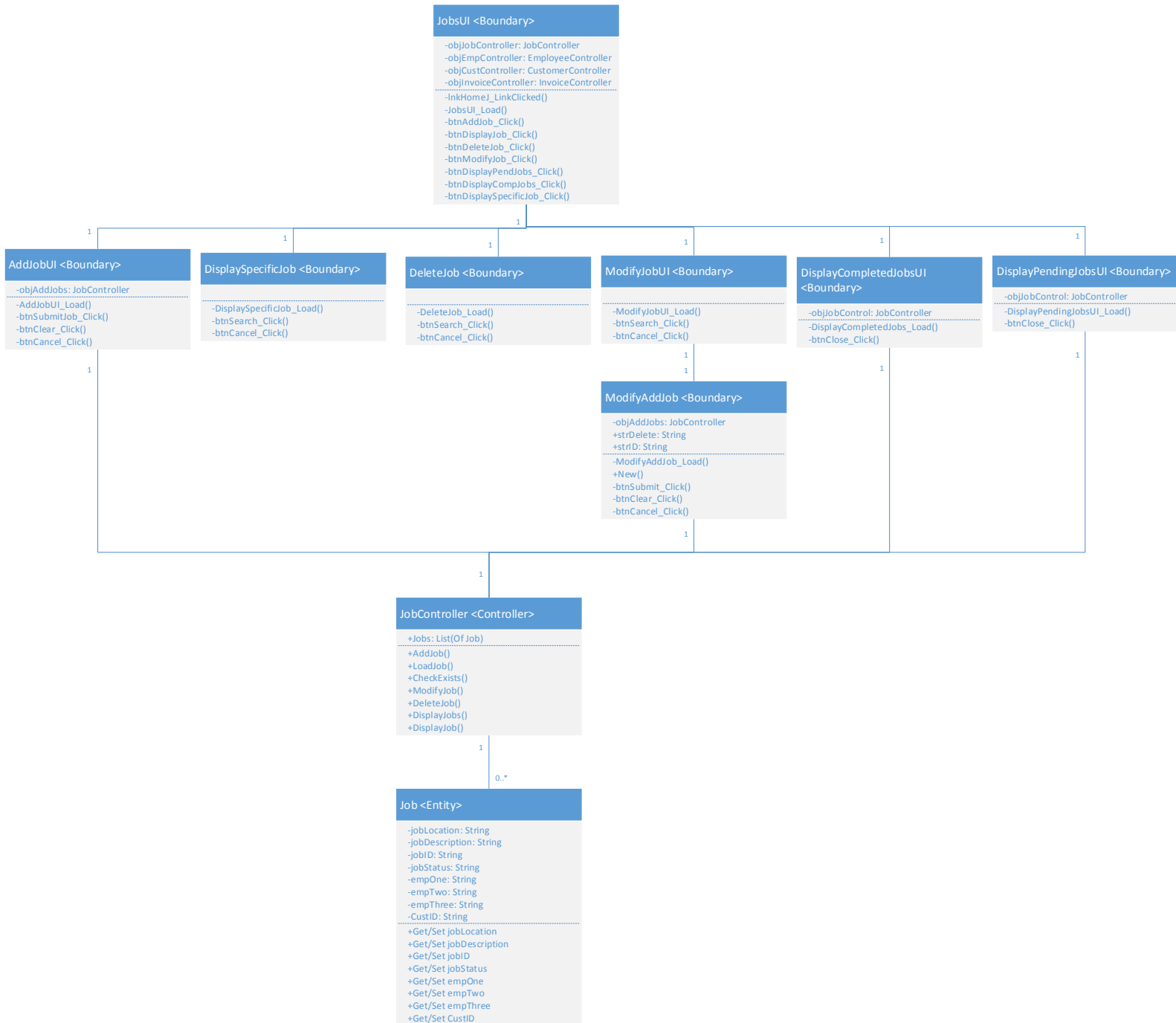
Customer Classes



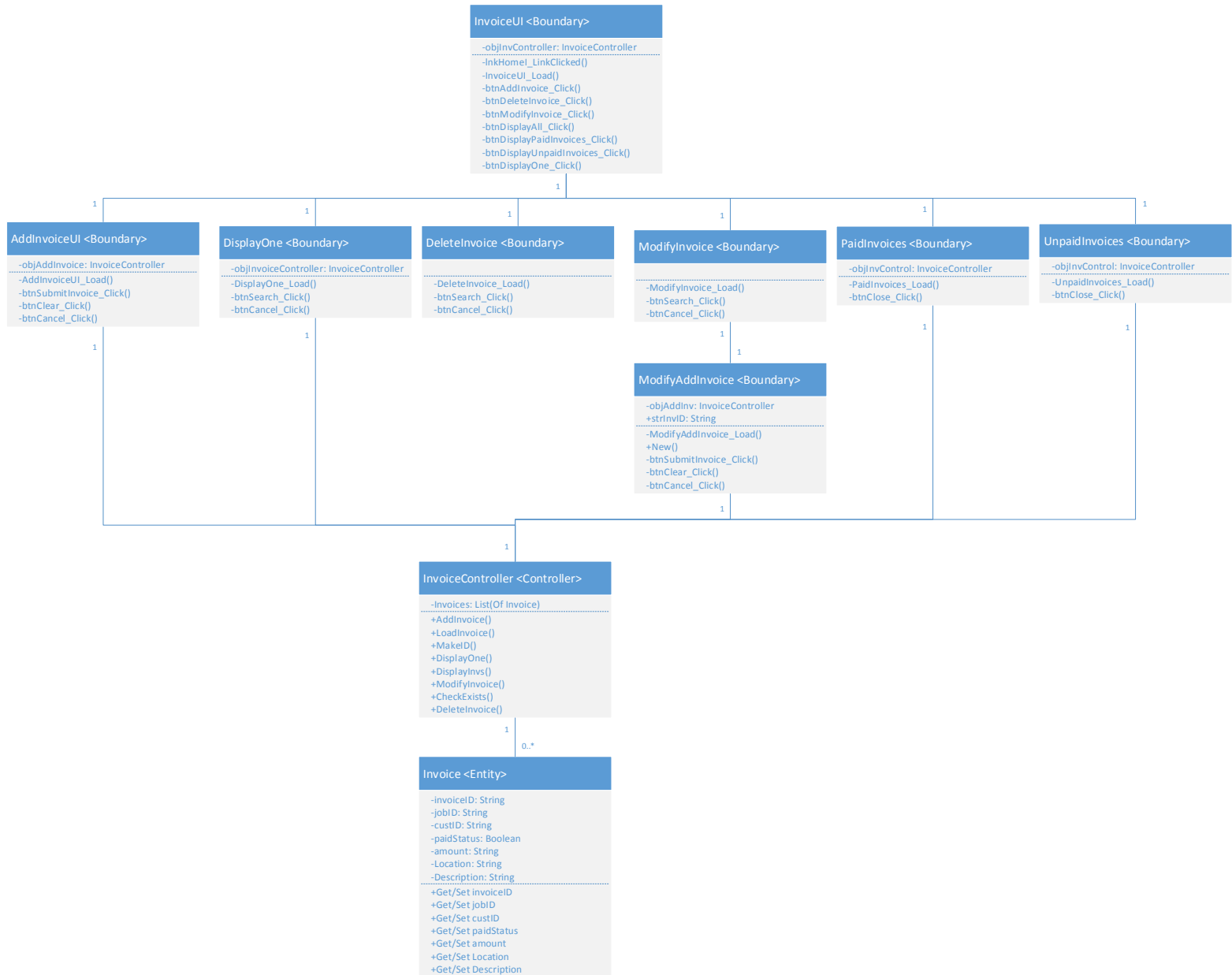
Employee Classes



Jobs Classes

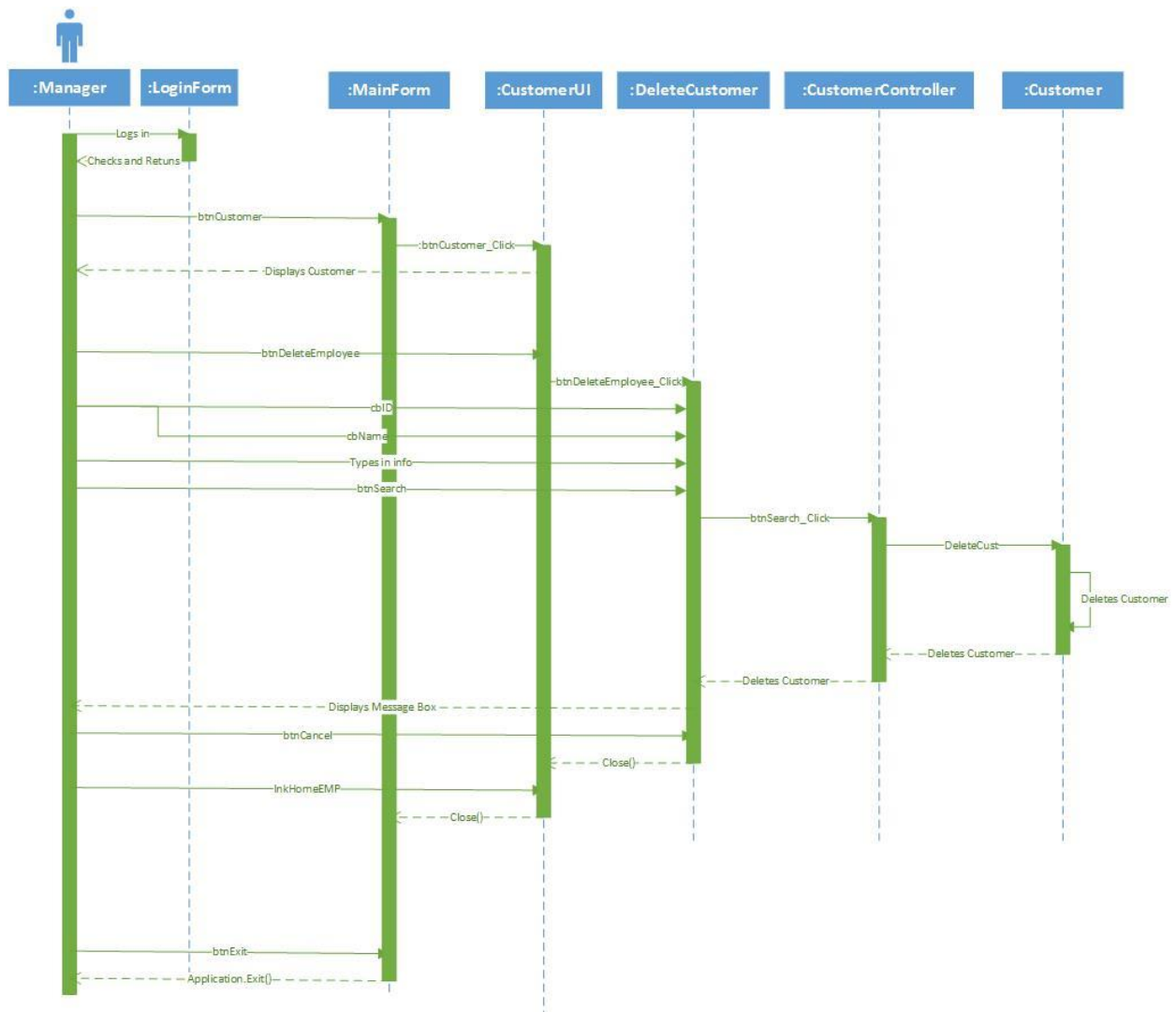


Invoice Classes

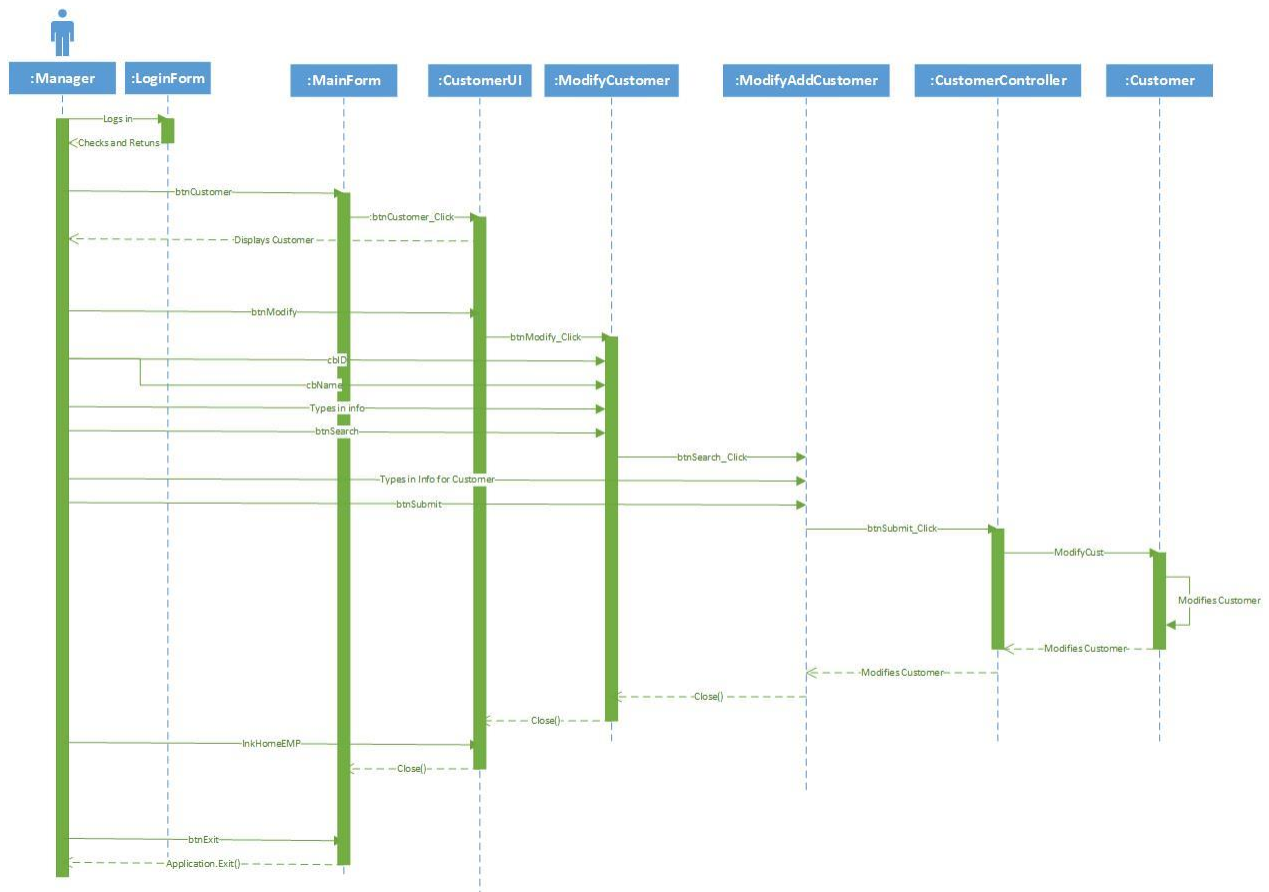


Interaction Diagrams

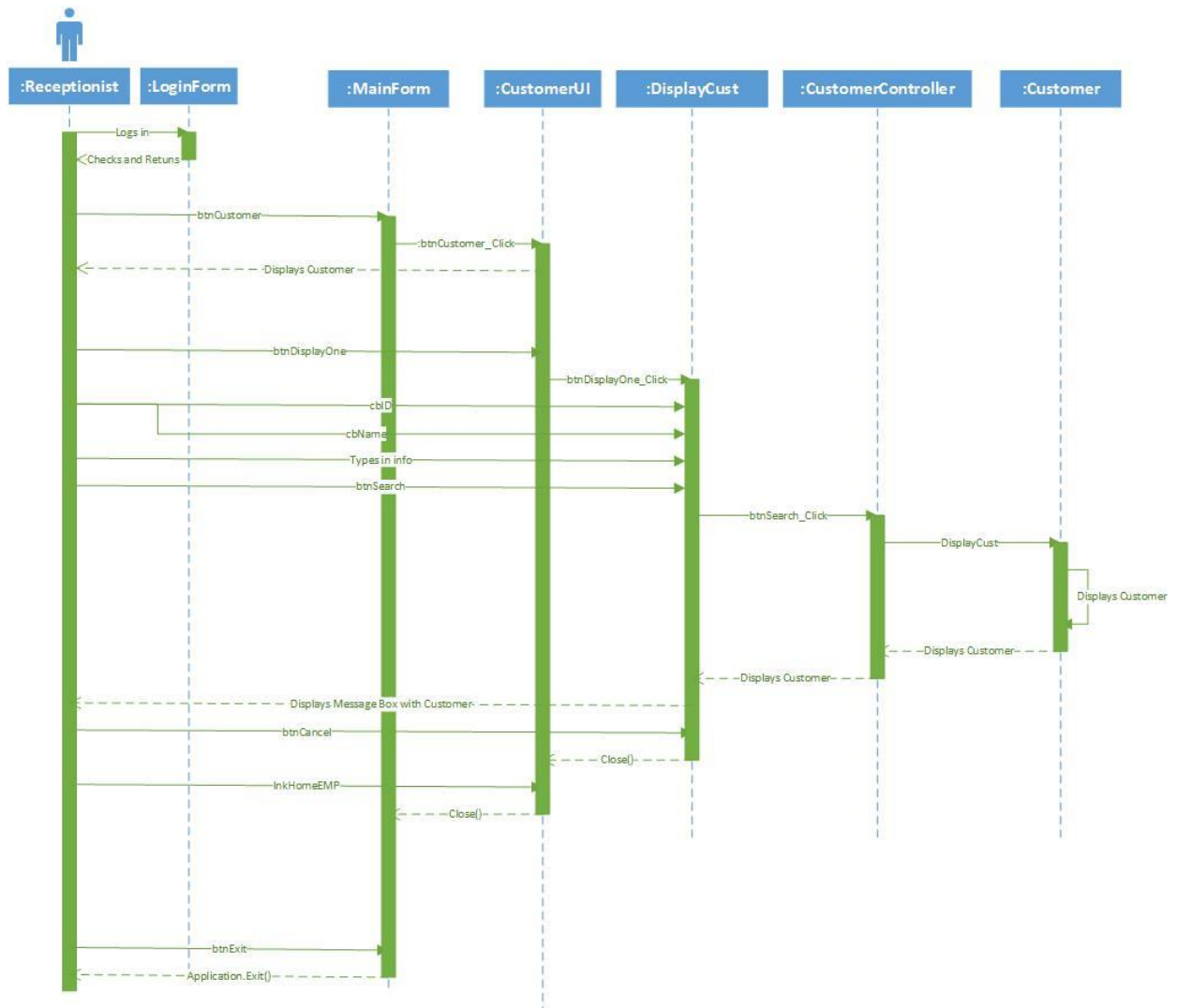
This is a detailed sequence diagram for 1b) Delete an existing customer



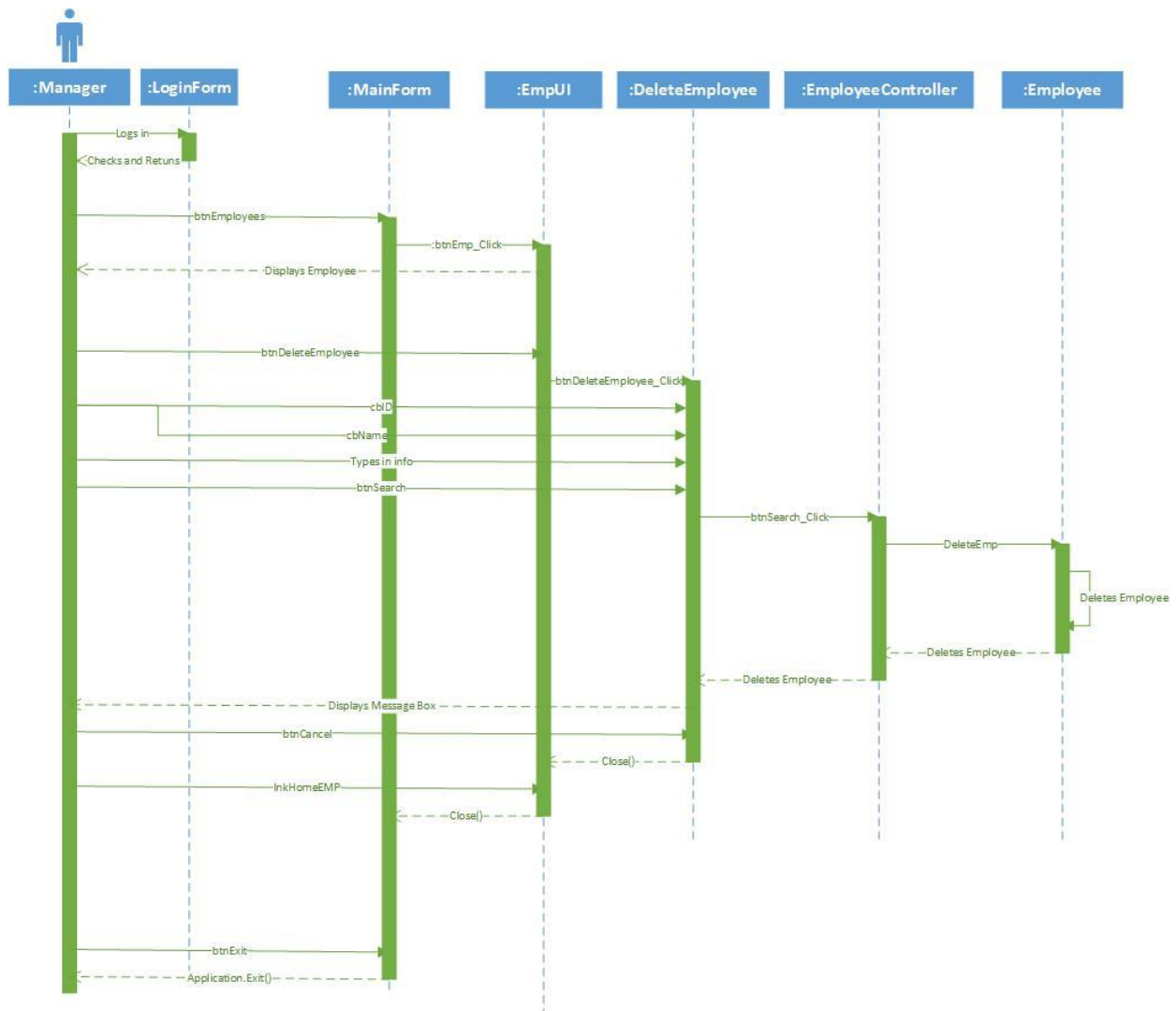
This is a detailed sequence diagram for 1c) Modify information about an existing customer



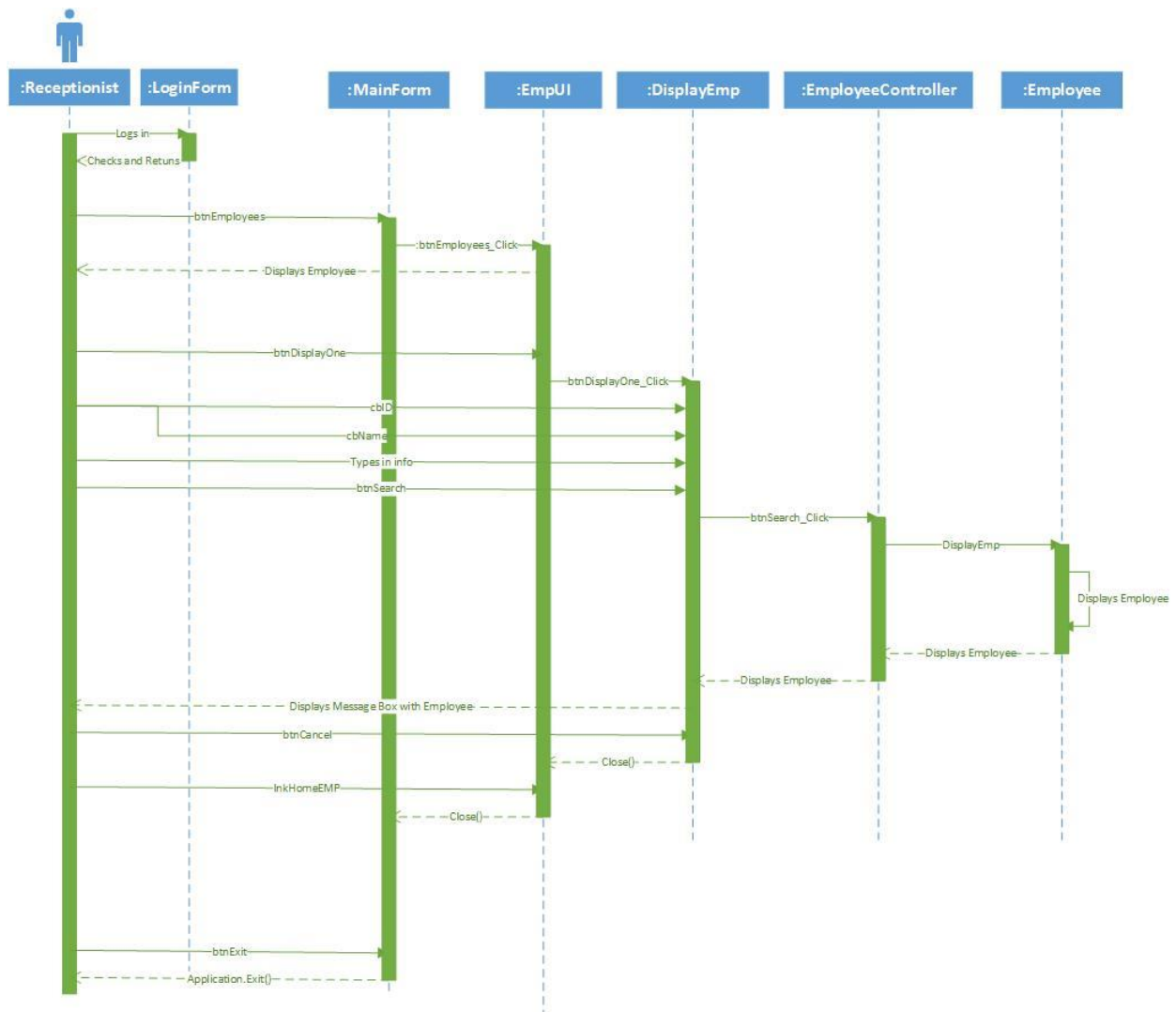
This is a detailed sequence diagram for 1d) Display/print information about an existing customer based of his/her id or name



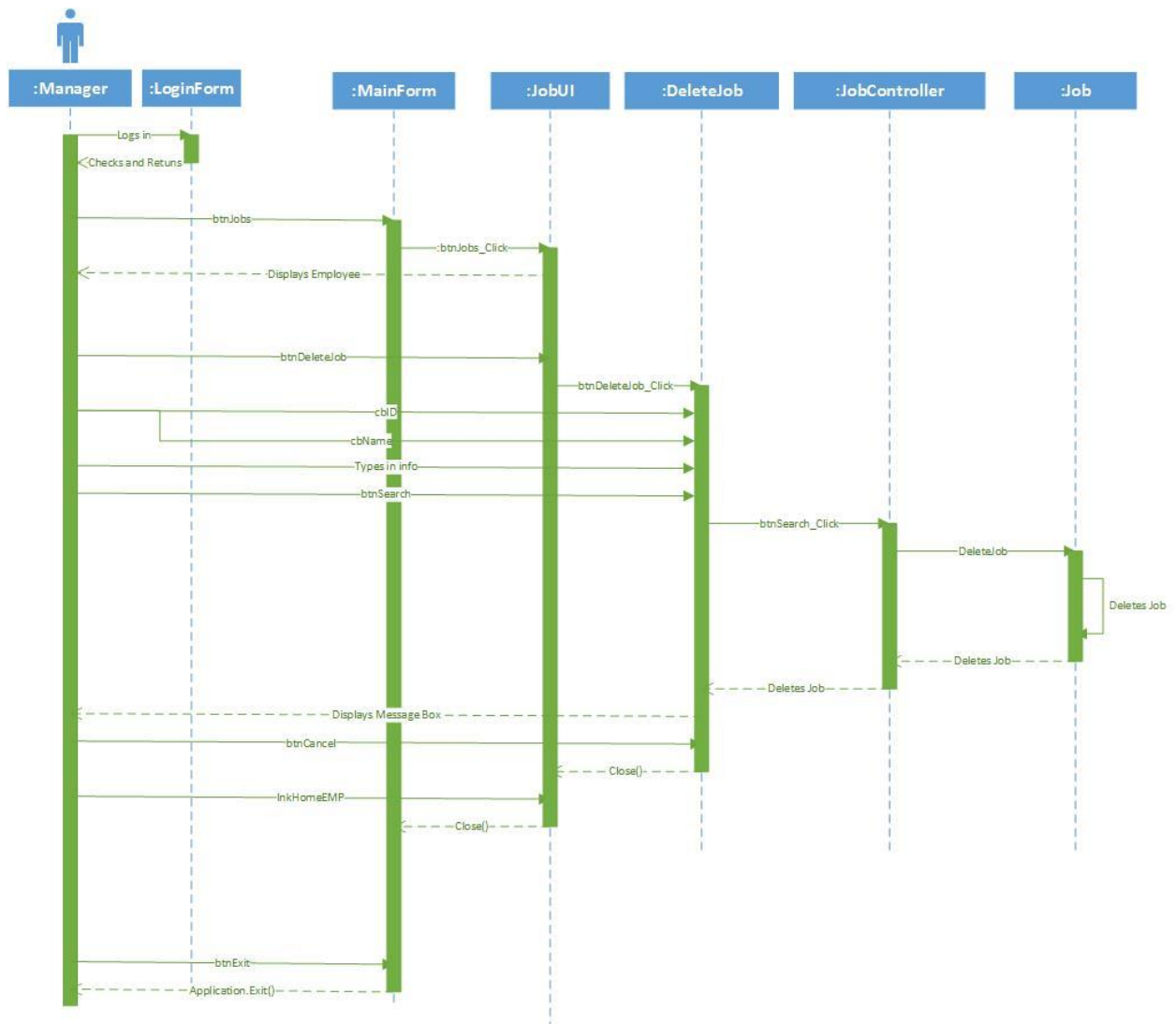
This is a detailed sequence diagram for 2b) Delete an existing employee



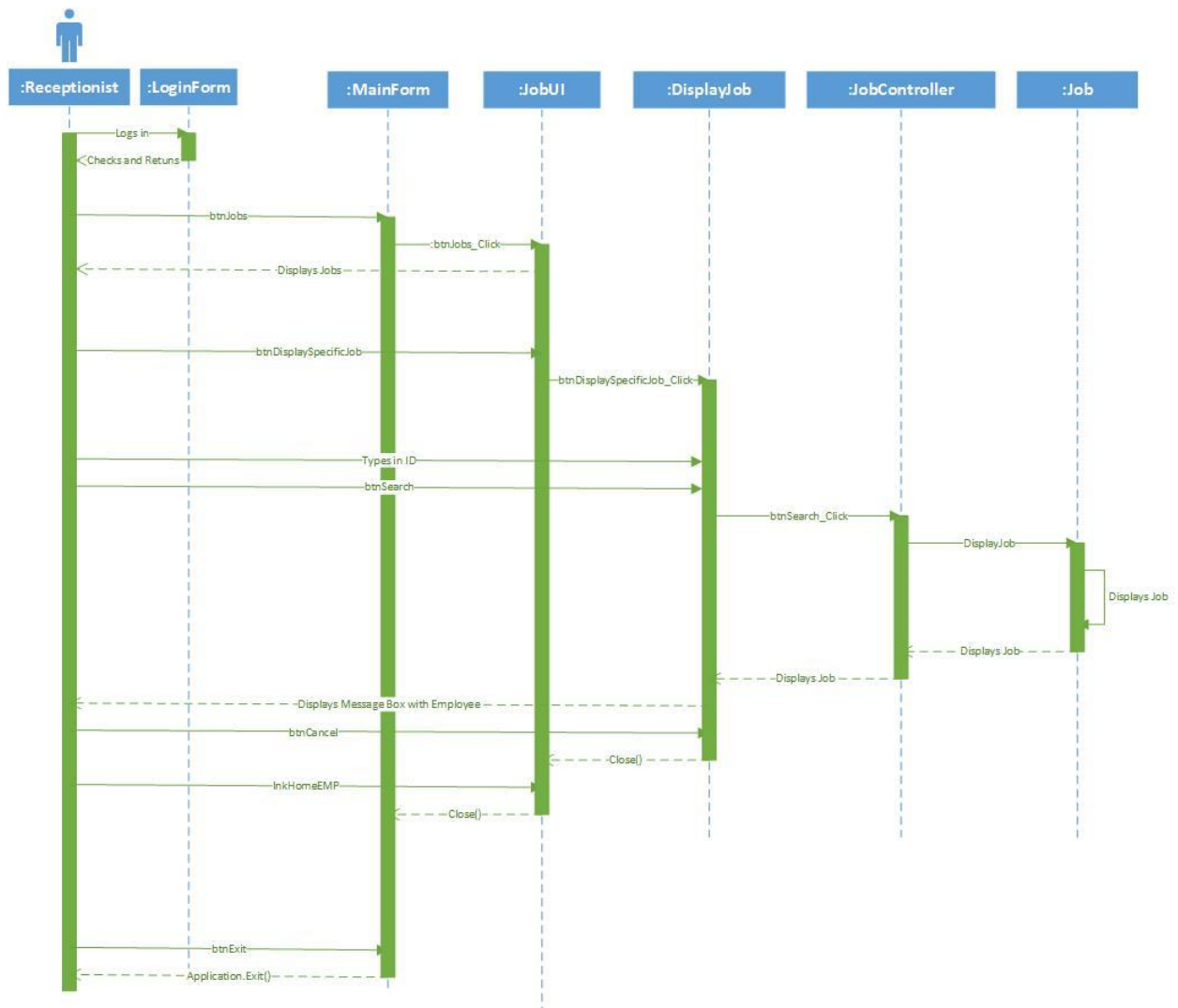
This is a detailed sequence diagram for 2c) Display/print information about an existing employee



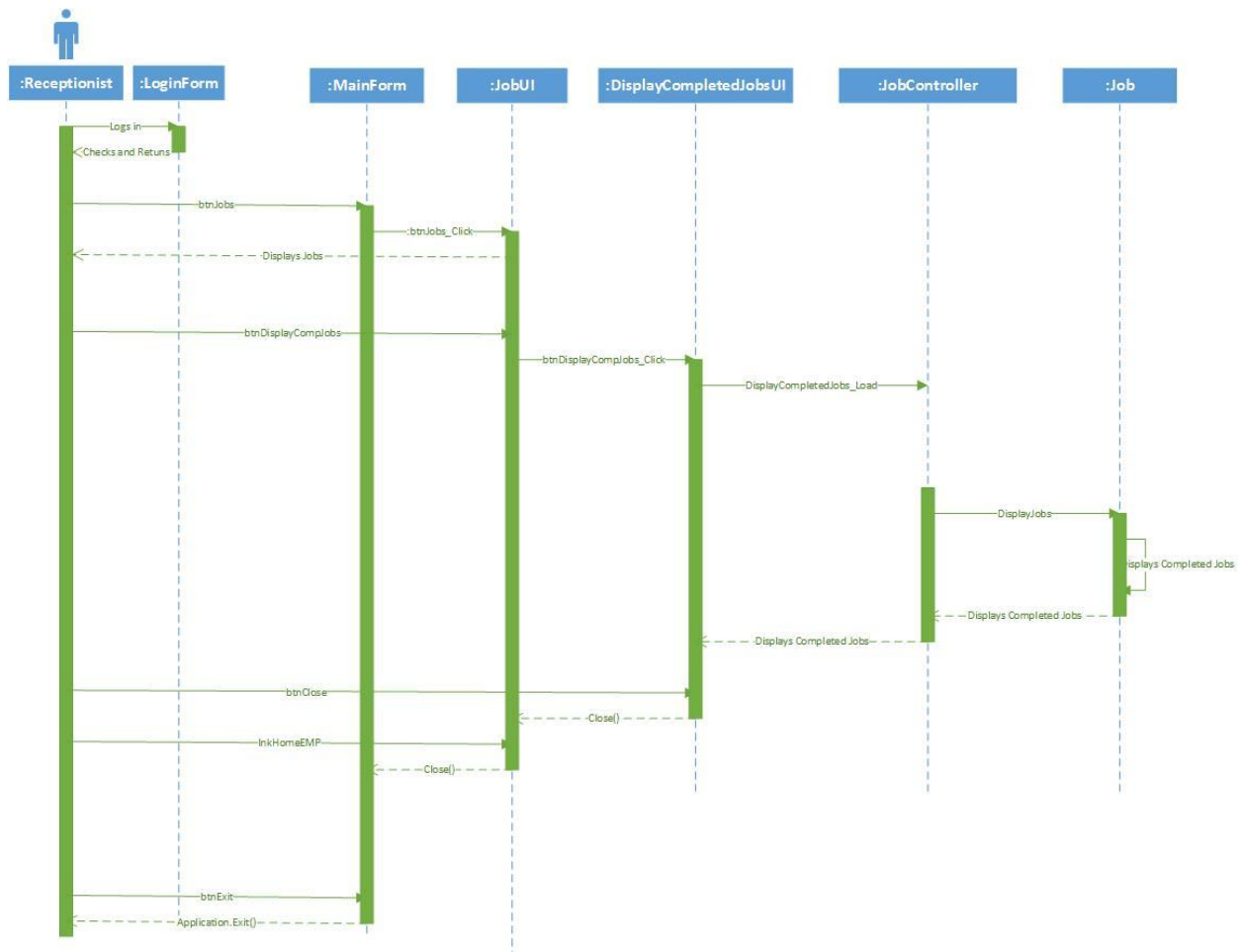
This is a detailed sequence diagram for 3b) Delete/cancel an existing job



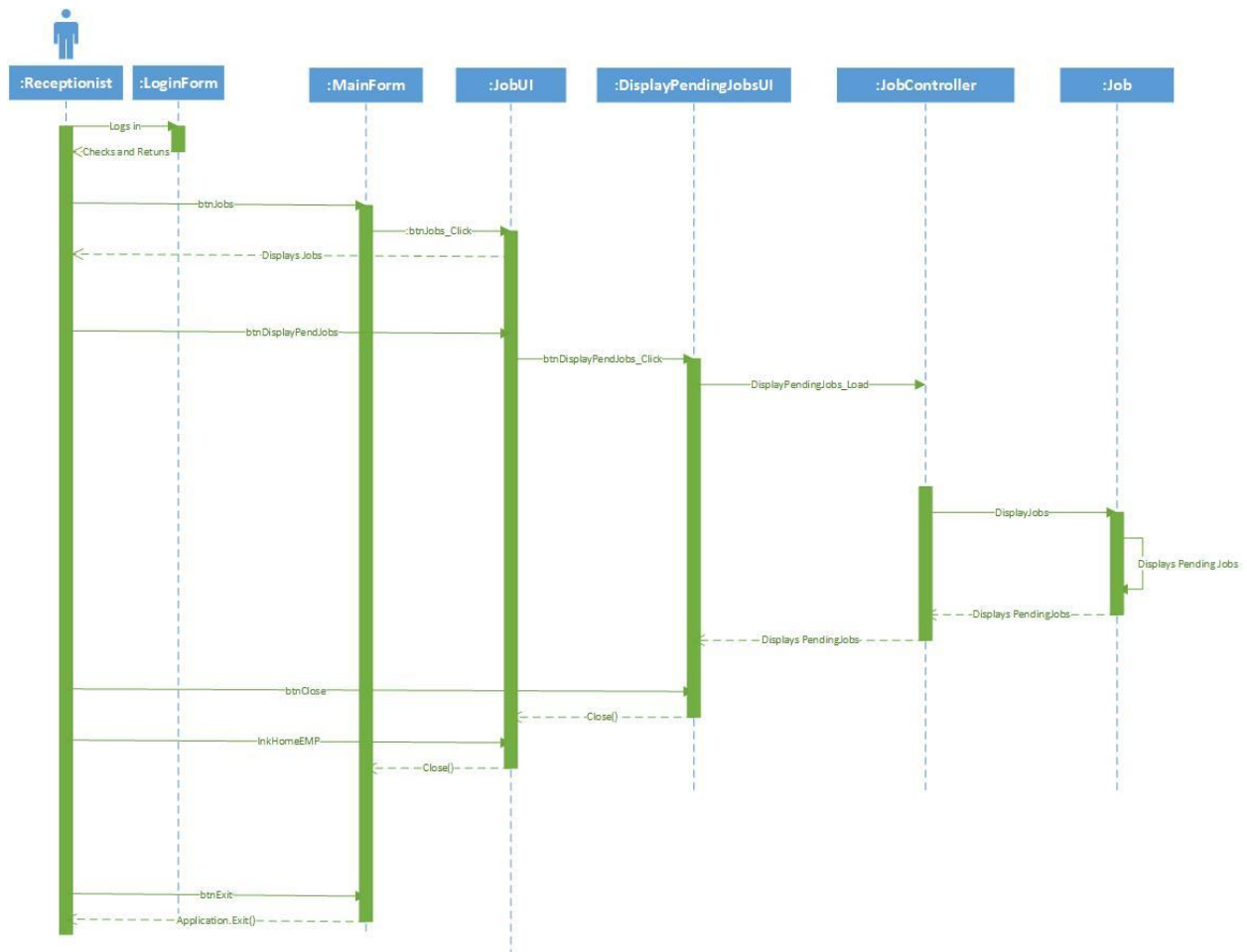
This is a detailed sequence diagram for 3c) Display/print the status of a specific job



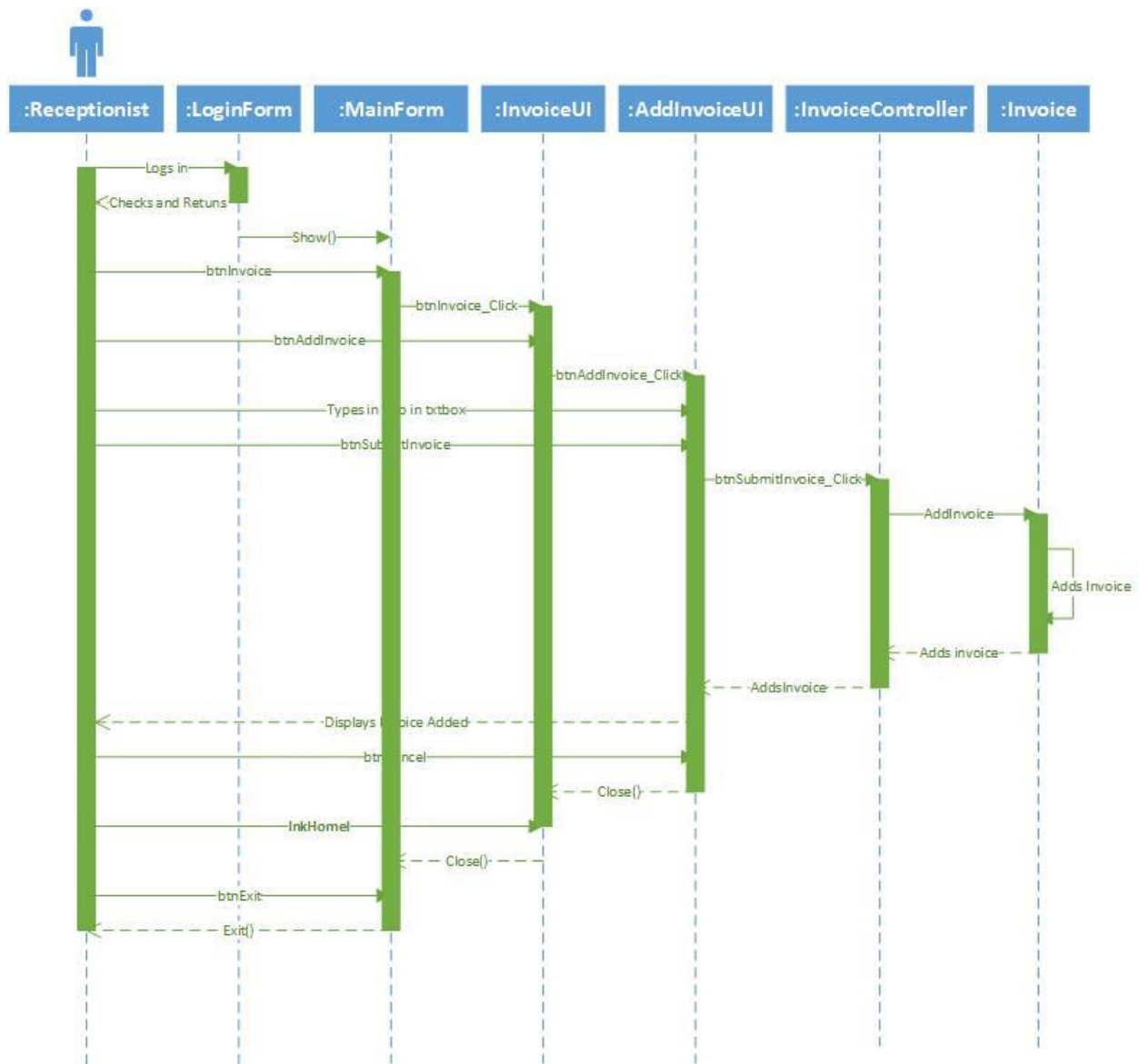
This is a detailed sequence diagram for 3e) Display a list of all completed jobs



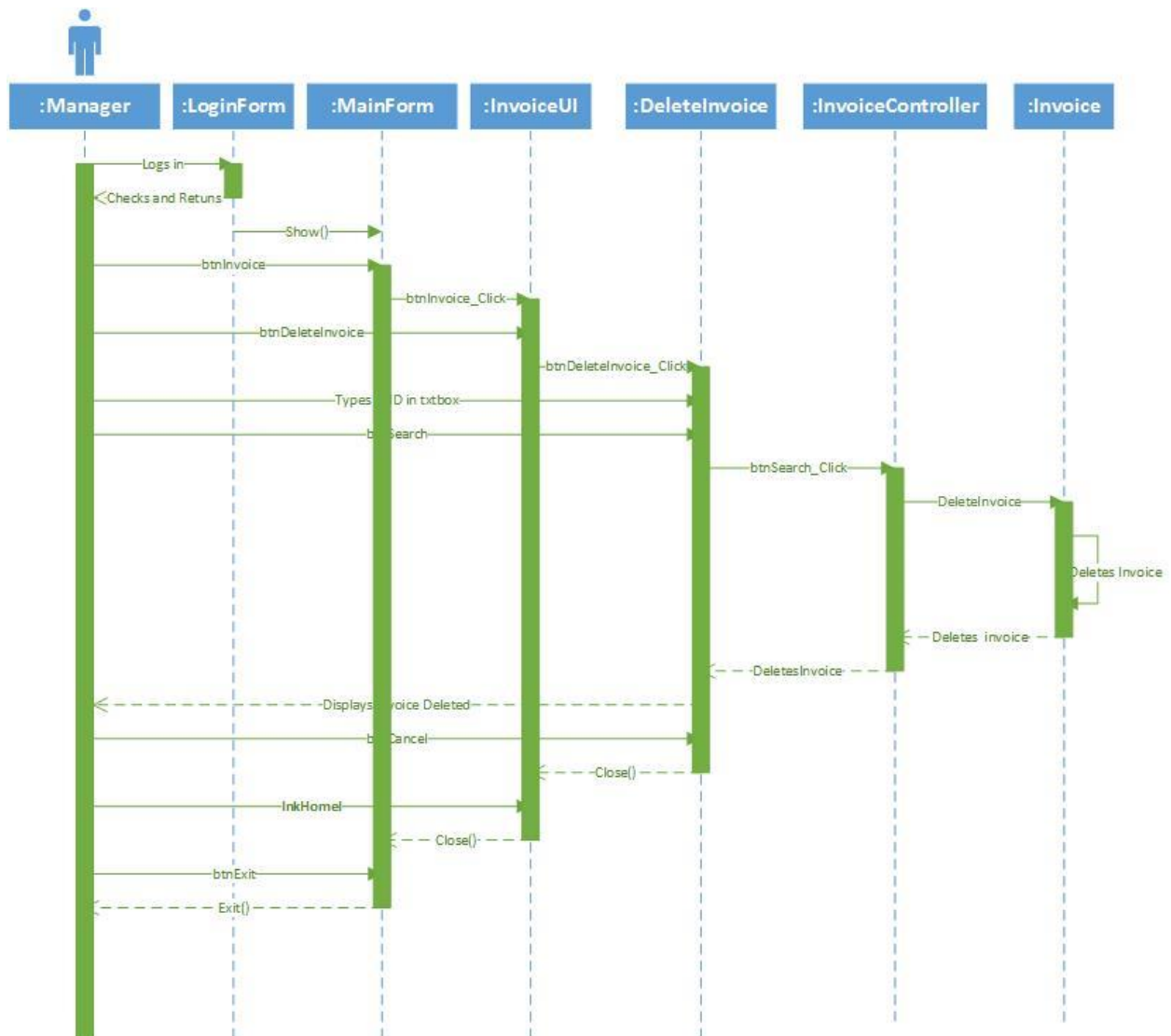
This is a detailed sequence diagram for 3f) Display a list of all pending jobs



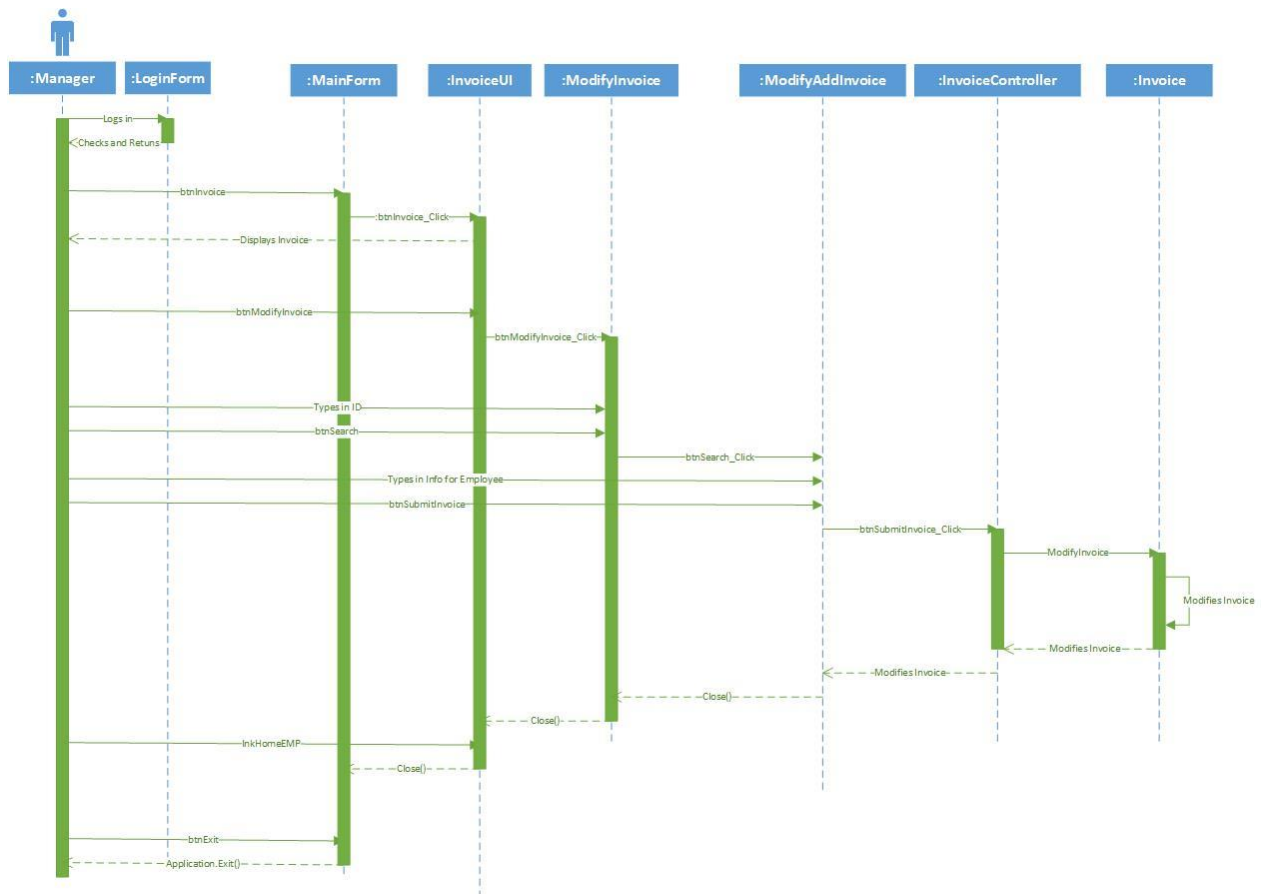
This is a detailed sequence diagram for 4a) Create a new invoice



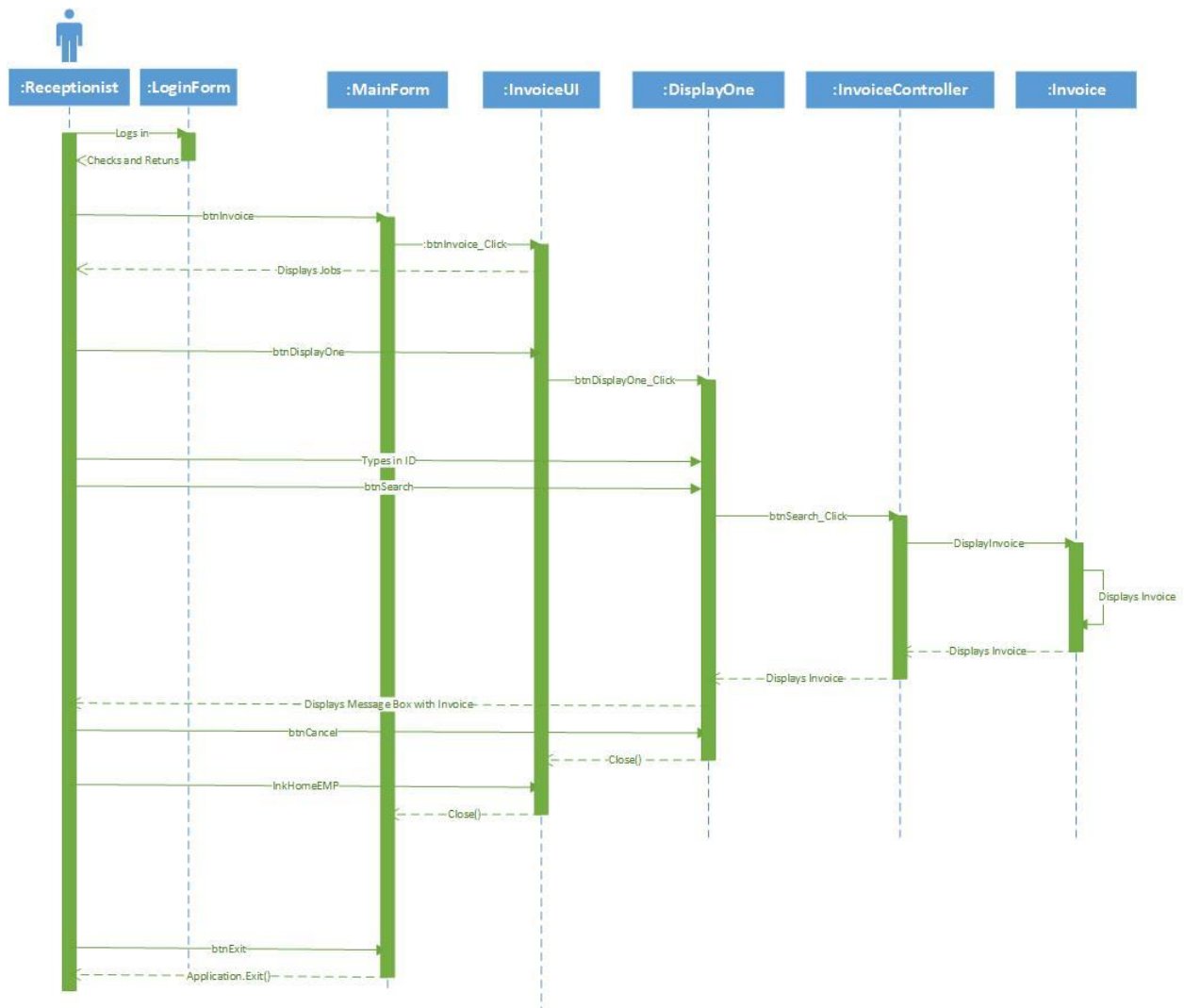
This is a detailed sequence diagram for 4b) Cancel an existing invoice



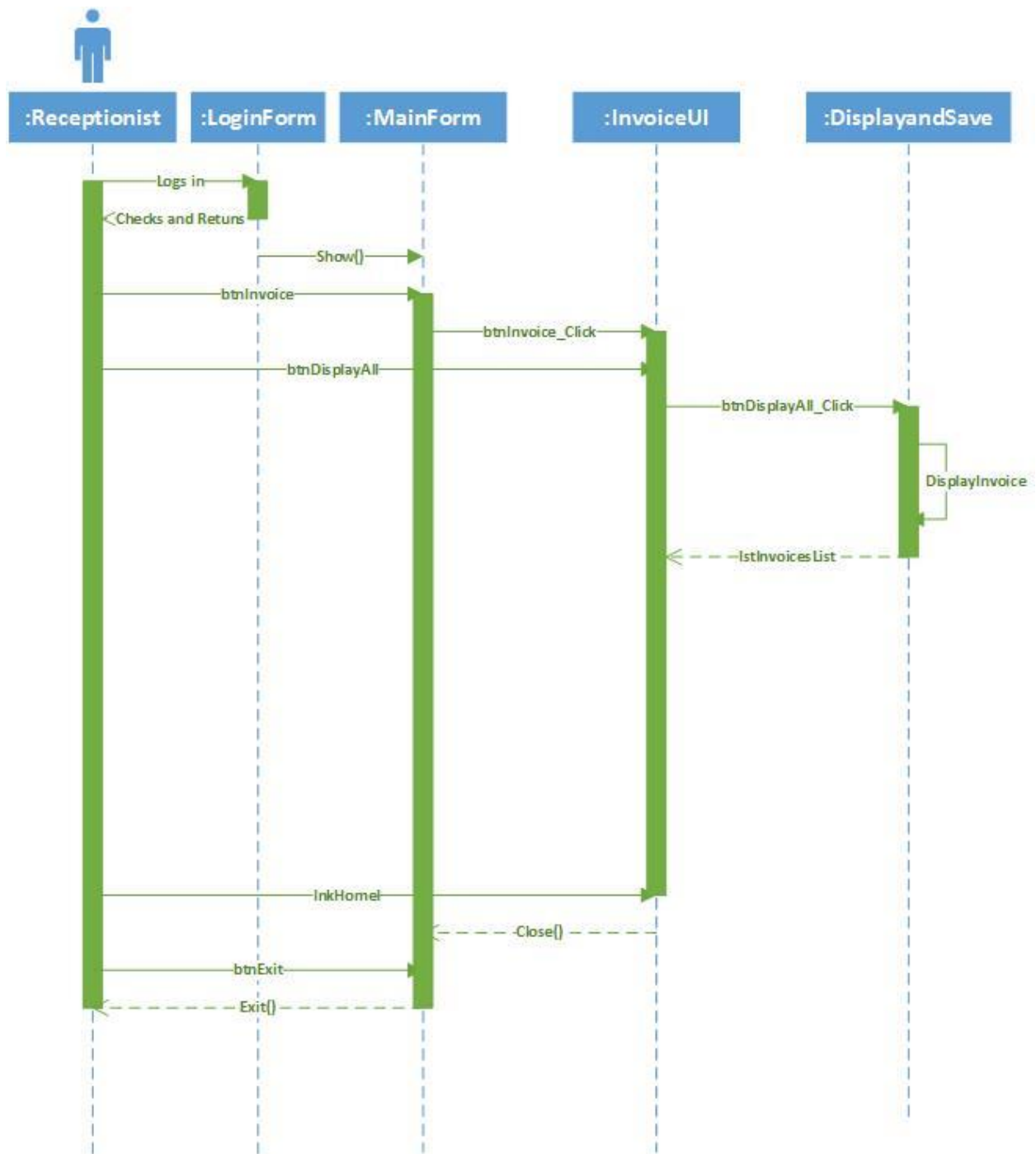
This is a detailed sequence diagram for 4c) Modify an existing invoice



This is a detailed sequence diagram for 4d) Display/print an existing invoice



This is a detailed sequence diagram for 4e) Display a list of all invoices and their payment status



Overall Design for GUI

Login Page

This is where a user logs in



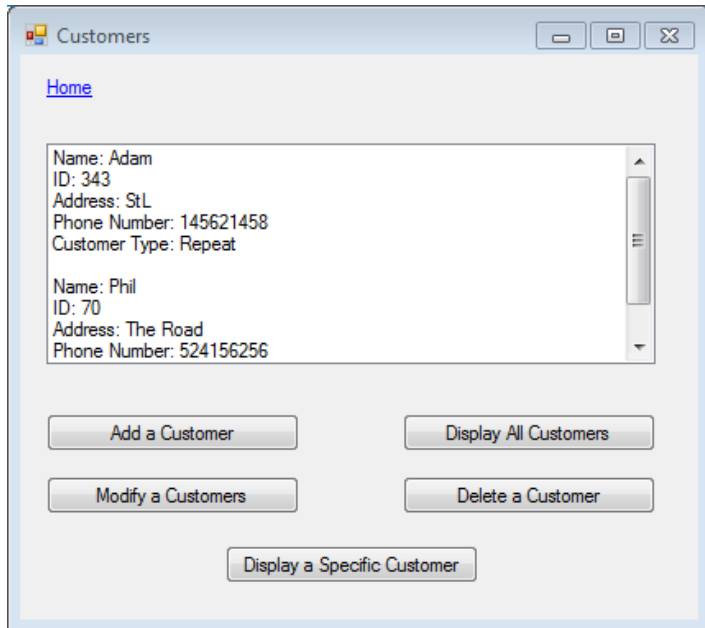
Home Page

This is where a user navigates to the various categories



Customers Page

This is where a user sees all the customers and performs various requirements to customers



The screenshot shows a window titled "Customers" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a blue underlined link labeled "Home". Below the link is a scrollable text area containing two customer records. The first record is for Adam (ID: 343, Address: StL, Phone Number: 145621458, Customer Type: Repeat). The second record is for Phil (ID: 70, Address: The Road, Phone Number: 524156256). Below the scrollable area are five buttons: "Add a Customer", "Display All Customers", "Modify a Customers", "Delete a Customer", and "Display a Specific Customer".

Customers

[Home](#)

Name: Adam
ID: 343
Address: StL
Phone Number: 145621458
Customer Type: Repeat

Name: Phil
ID: 70
Address: The Road
Phone Number: 524156256

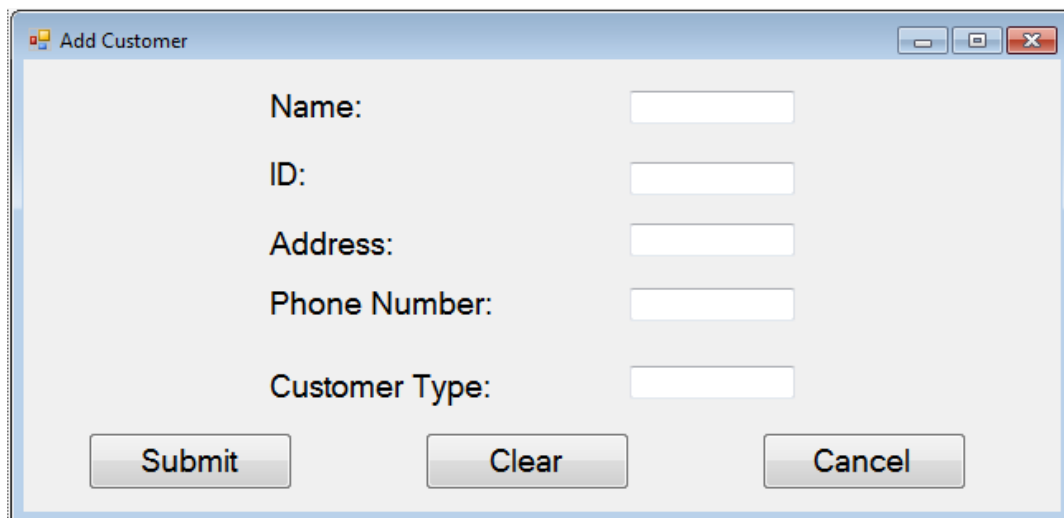
Add a Customer Display All Customers

Modify a Customers Delete a Customer

Display a Specific Customer

Adds Customer Page

This is the form to add a customer



The screenshot shows a window titled "Add Customer" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are five labels with corresponding text input fields: "Name:", "ID:", "Address:", "Phone Number:", and "Customer Type:". At the bottom of the window are three buttons: "Submit", "Clear", and "Cancel".

Add Customer

Name:

ID:

Address:

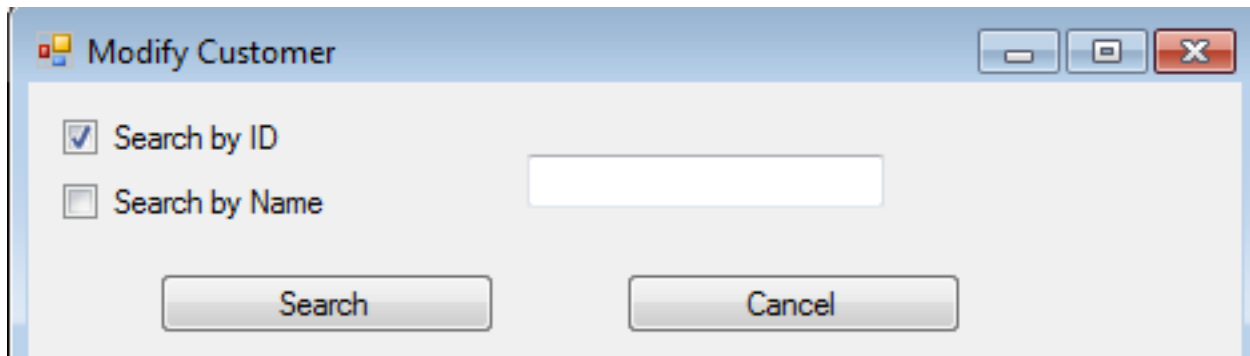
Phone Number:

Customer Type:

Submit Clear Cancel

Modify Customer Search Page

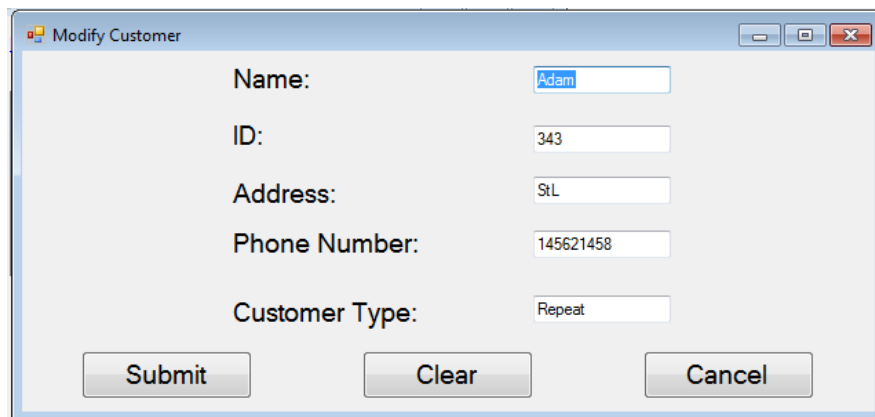
This is where a user searches for the customer he wants to modify



A screenshot of a web application window titled "Modify Customer". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a text input field. Below the input field are two buttons: "Search" and "Cancel".

Modify a Customer Page

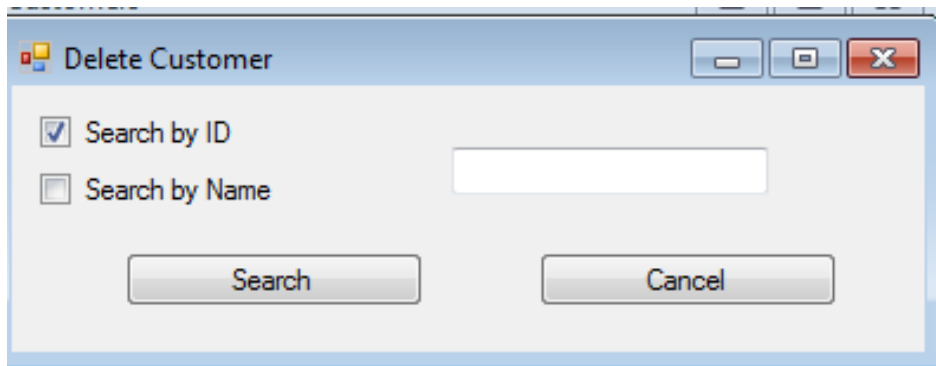
This is where the user modifies a customer's info



A screenshot of a web application window titled "Modify Customer". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are five labeled text input fields arranged vertically: "Name:" (containing "Adam"), "ID:" (containing "343"), "Address:" (containing "StL"), "Phone Number:" (containing "145621458"), and "Customer Type:" (containing "Repeat"). At the bottom of the window are three buttons: "Submit", "Clear", and "Cancel".

Delete Customer Search Page

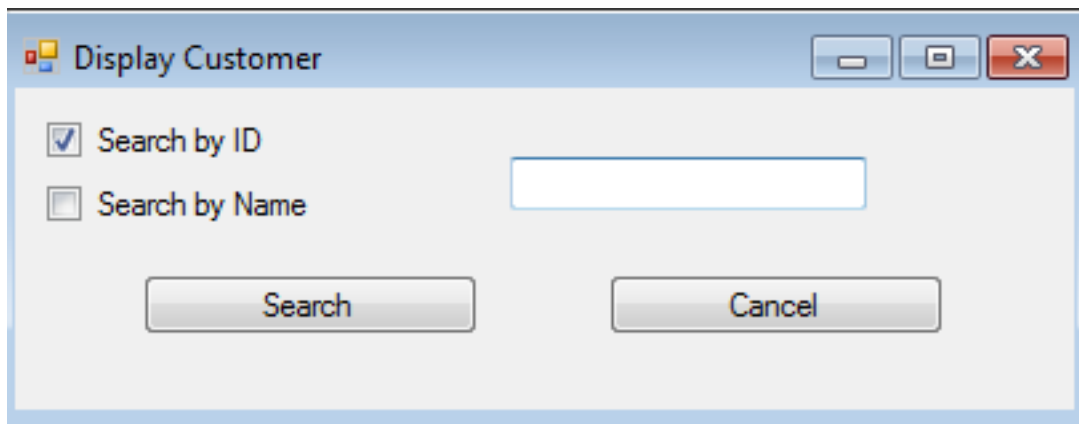
This is where the user searches for the customer they want to delete



A screenshot of a Windows-style dialog box titled "Delete Customer". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a white text input field. At the bottom of the dialog are two buttons: "Search" and "Cancel".

Display Customer Search Page

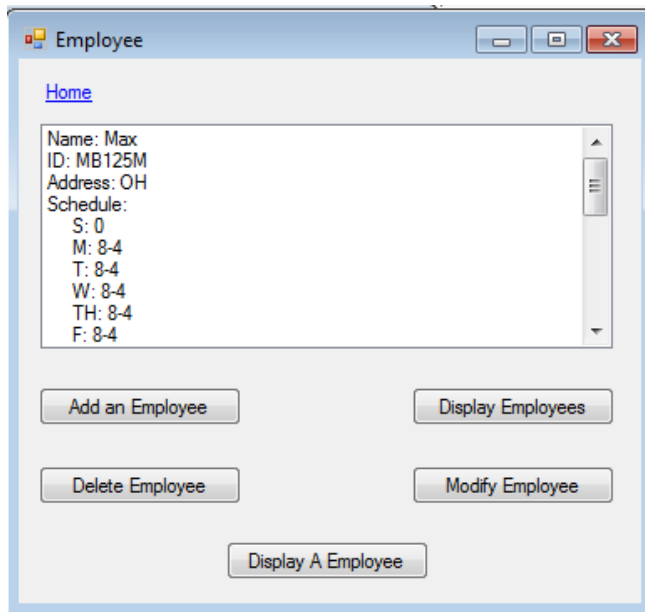
This is where a user searches for a specific customer to display



A screenshot of a Windows-style dialog box titled "Display Customer". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a white text input field. At the bottom of the dialog are two buttons: "Search" and "Cancel".

Employee Page

This is where a user sees all the employees and performs various requirements to employees



The 'Employee' window displays a list of employee details in a text area. The details shown are: Name: Max, ID: MB125M, Address: OH, and Schedule: S: 0, M: 8-4, T: 8-4, W: 8-4, TH: 8-4, F: 8-4. Below the text area are five buttons: 'Add an Employee', 'Display Employees', 'Delete Employee', 'Modify Employee', and 'Display A Employee'.

Employee

[Home](#)

Name: Max
ID: MB125M
Address: OH
Schedule:
S: 0
M: 8-4
T: 8-4
W: 8-4
TH: 8-4
F: 8-4

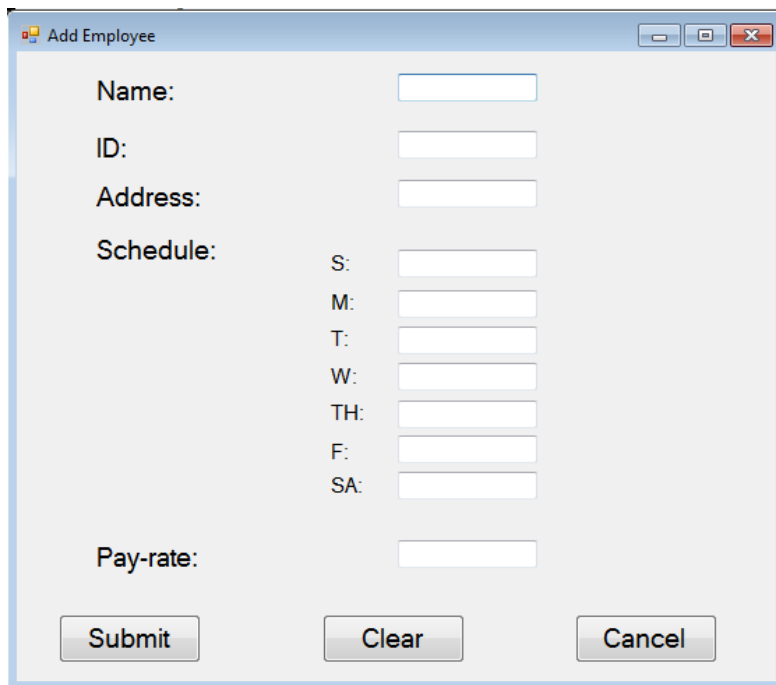
Add an Employee Display Employees

Delete Employee Modify Employee

Display A Employee

Add Employee Page

This is where a user adds an employee



The 'Add Employee' window contains input fields for Name, ID, Address, and Pay-rate. The Schedule section includes individual input fields for S, M, T, W, TH, F, and SA. At the bottom are three buttons: 'Submit', 'Clear', and 'Cancel'.

Add Employee

Name:

ID:

Address:

Schedule:

S:

M:

T:

W:

TH:

F:

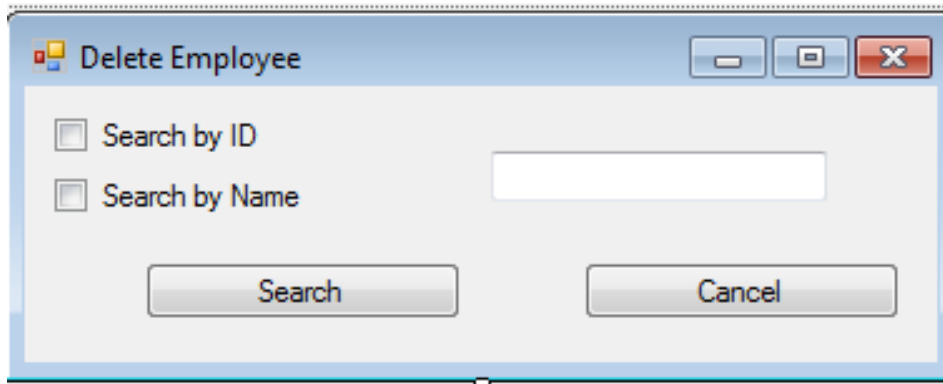
SA:

Pay-rate:

Submit Clear Cancel

Delete Employee Search Page

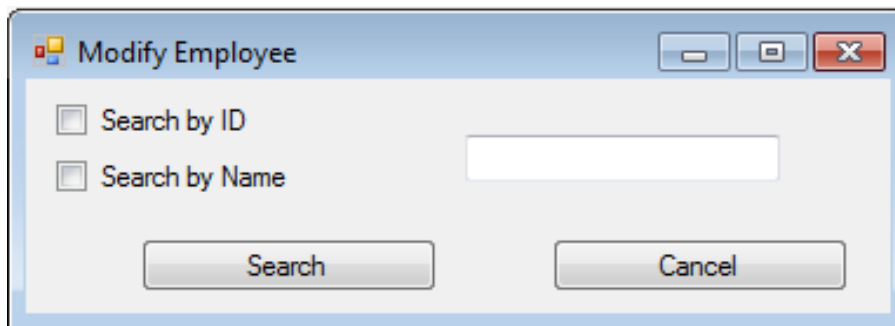
This is where a user searches for an employee to delete



A screenshot of a Windows-style dialog box titled "Delete Employee". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" and "Search by Name". To the right of these is a text input field. At the bottom, there are two buttons: "Search" and "Cancel".

Modify Employee Search Page

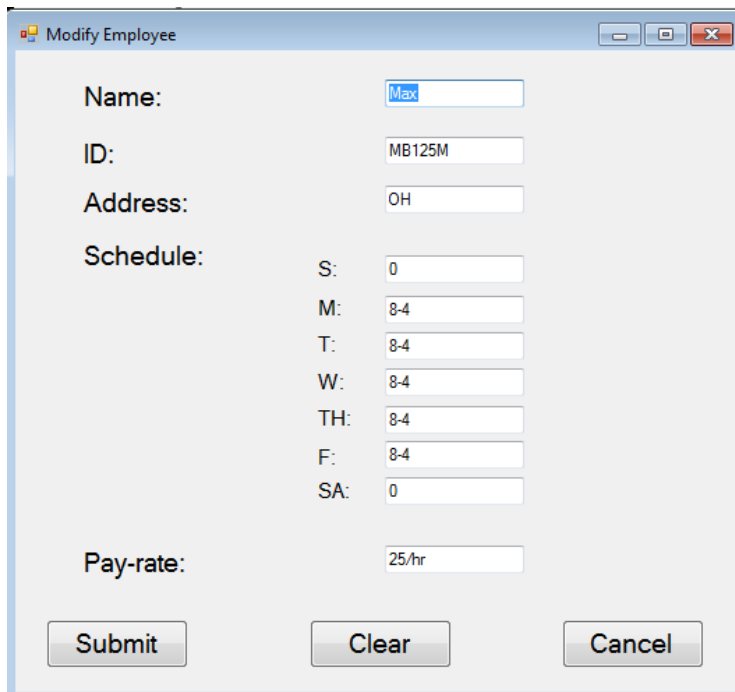
This is where a user searches for an employee to delete



A screenshot of a Windows-style dialog box titled "Modify Employee". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" and "Search by Name". To the right of these is a text input field. At the bottom, there are two buttons: "Search" and "Cancel".

Modify Employee Page

This is where a user modifies an employee's info

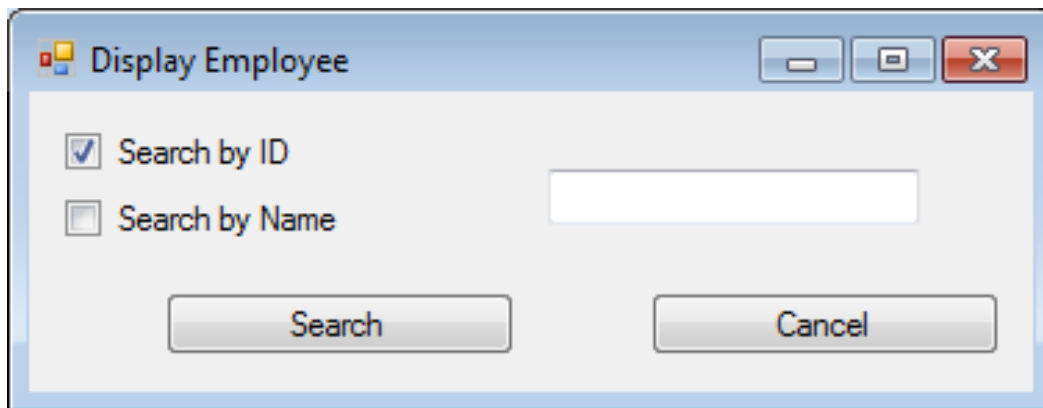


The 'Modify Employee' dialog box contains the following fields and controls:

- Name:** Text input field containing 'Max'.
- ID:** Text input field containing 'MB125M'.
- Address:** Text input field containing 'OH'.
- Schedule:** A group of seven text input fields for days of the week:
 - S: 0
 - M: 8-4
 - T: 8-4
 - W: 8-4
 - TH: 8-4
 - F: 8-4
 - SA: 0
- Pay-rate:** Text input field containing '25/hr'.
- Buttons:** 'Submit', 'Clear', and 'Cancel' buttons at the bottom.

Display an Employee Search Page

This is where the user searches for an employee to display

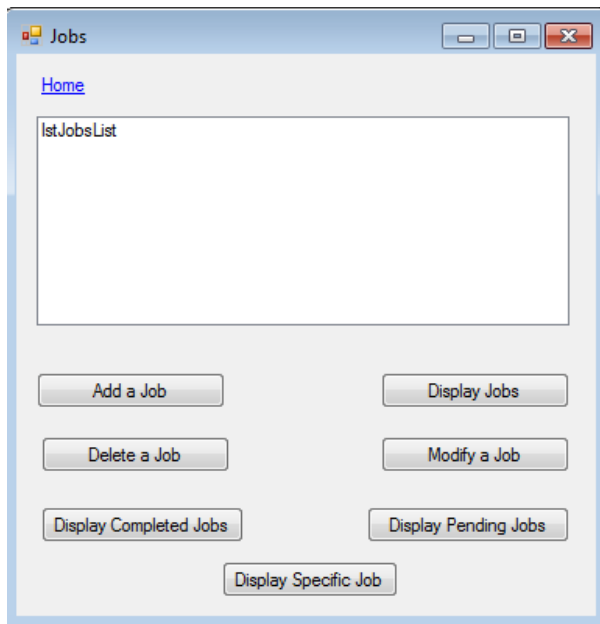


The 'Display Employee' dialog box contains the following fields and controls:

- Search by ID:** A checked checkbox.
- Search by Name:** An unchecked checkbox.
- Search Input:** A text input field for the search criteria.
- Buttons:** 'Search' and 'Cancel' buttons at the bottom.

Jobs Page

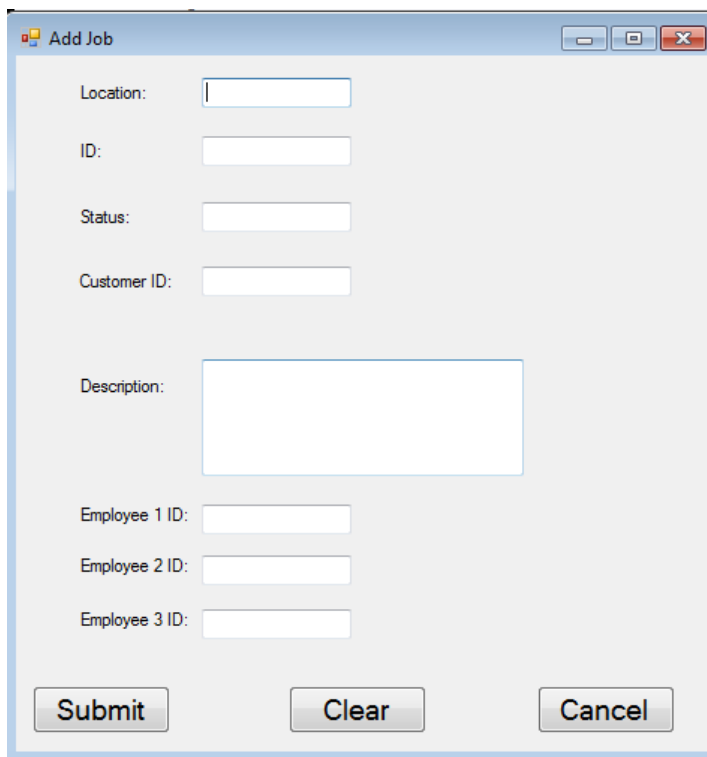
This is where a user sees all the jobs and performs various requirements to jobs



The screenshot shows a window titled "Jobs" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a link labeled "Home" at the top left. Below the link is a large, empty rectangular box labeled "lstJobsList". At the bottom of the window, there are six buttons arranged in three rows: "Add a Job", "Delete a Job", and "Display Completed Jobs" in the first column; "Display Jobs", "Modify a Job", and "Display Pending Jobs" in the second column; and "Display Specific Job" centered at the bottom.

Add a Job Page

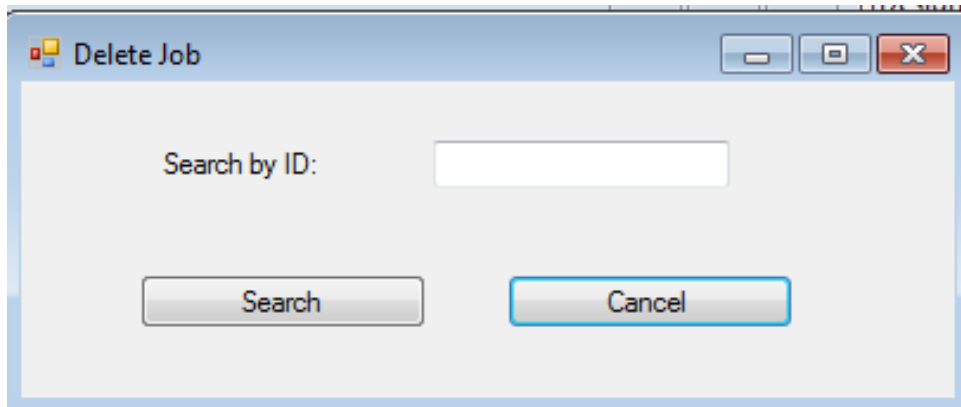
This is where a user adds a job



The screenshot shows a window titled "Add Job" with a standard Windows-style title bar. The window contains several input fields for job details: "Location:" with a single-line text box, "ID:" with a single-line text box, "Status:" with a single-line text box, "Customer ID:" with a single-line text box, "Description:" with a multi-line text box, "Employee 1 ID:" with a single-line text box, "Employee 2 ID:" with a single-line text box, and "Employee 3 ID:" with a single-line text box. At the bottom of the window, there are three buttons: "Submit", "Clear", and "Cancel".

Delete Job Search

This is where a user searches for a job to delete



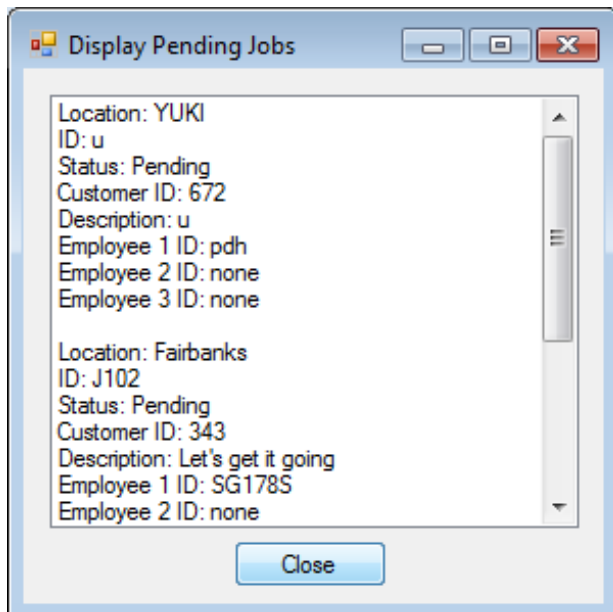
Display Completed Jobs Page

This is where a user looks at the completed jobs



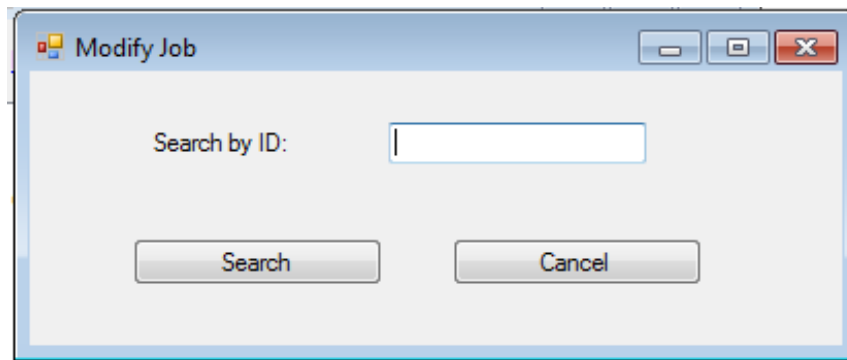
Pending Jobs Page

This is where a user looks at jobs not completed



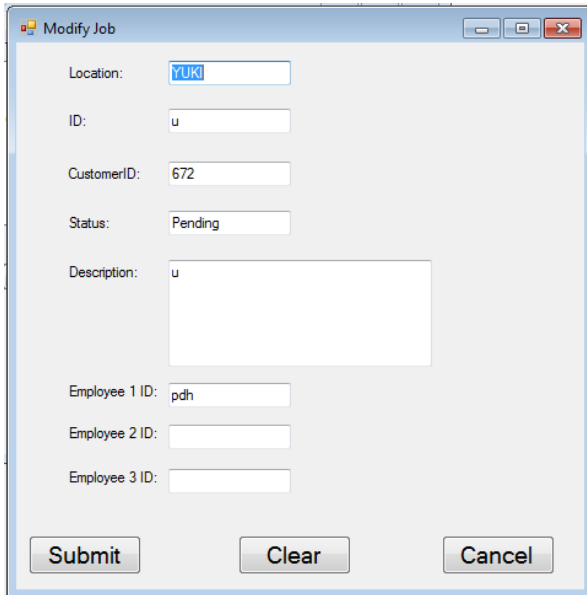
Modify a Job Search Page

This is where a user goes to search for a job to modify



Modify a Job Page

This is where a user modifies a job

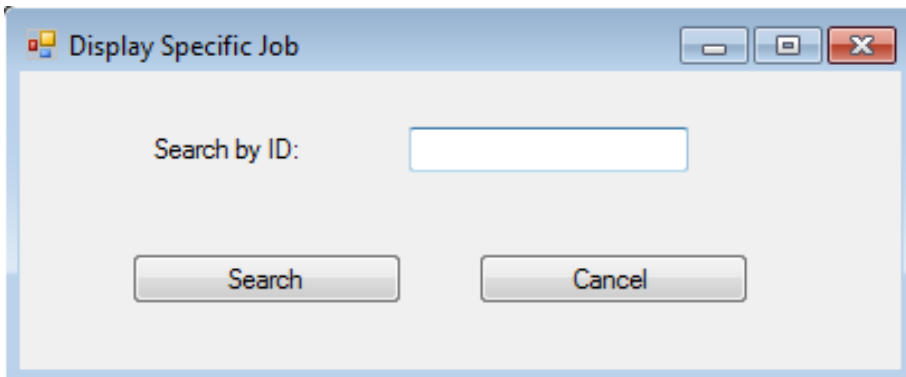


The 'Modify Job' dialog box contains the following fields and controls:

- Location:
- ID:
- CustomerID:
- Status:
- Description:
- Employee 1 ID:
- Employee 2 ID:
- Employee 3 ID:
- Buttons: Submit, Clear, Cancel

Display a Specific Job Search Page

This is where a user searches for a specific job to display

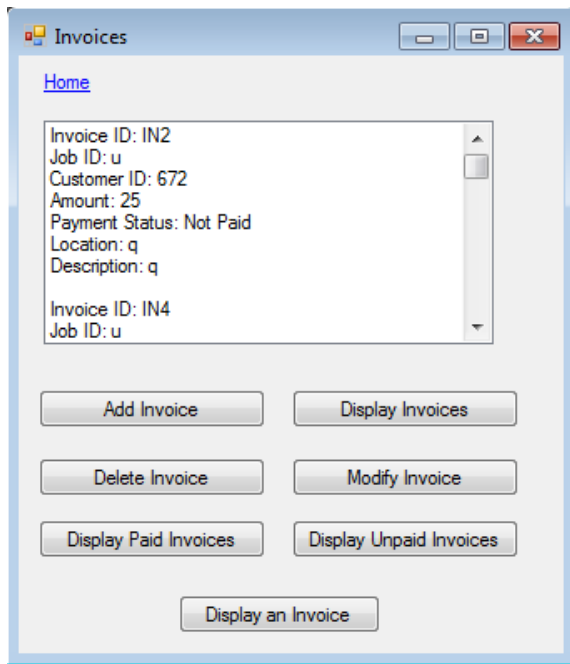


The 'Display Specific Job' dialog box contains the following fields and controls:

- Search by ID:
- Buttons: Search, Cancel

Invoice Page

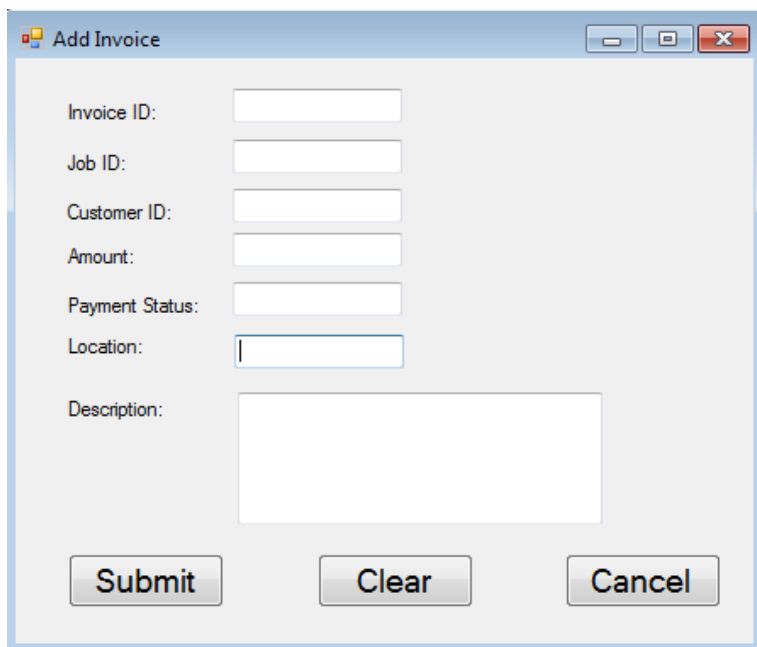
This is where a user sees all the invoices and performs various requirements to invoices



The 'Invoices' window displays a list of invoices in a scrollable area. The first invoice has the following details: Invoice ID: IN2, Job ID: u, Customer ID: 672, Amount: 25, Payment Status: Not Paid, Location: q, and Description: q. The second invoice shows Invoice ID: IN4 and Job ID: u. Below the list are six buttons: 'Add Invoice', 'Display Invoices', 'Delete Invoice', 'Modify Invoice', 'Display Paid Invoices', and 'Display Unpaid Invoices'. A 'Display an Invoice' button is located at the bottom center.

Add an Invoice Page

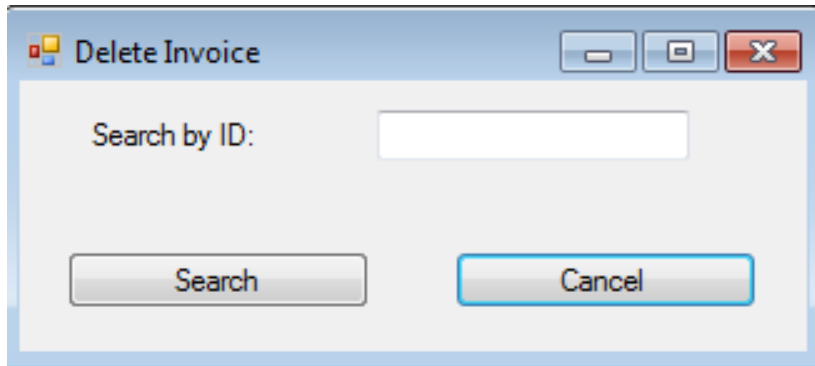
This is where a user adds an invoice



The 'Add Invoice' window contains input fields for the following fields: Invoice ID, Job ID, Customer ID, Amount, Payment Status, Location, and Description. The Description field is a larger text area. At the bottom of the window are three buttons: 'Submit', 'Clear', and 'Cancel'.

Delete an Invoice Search Page

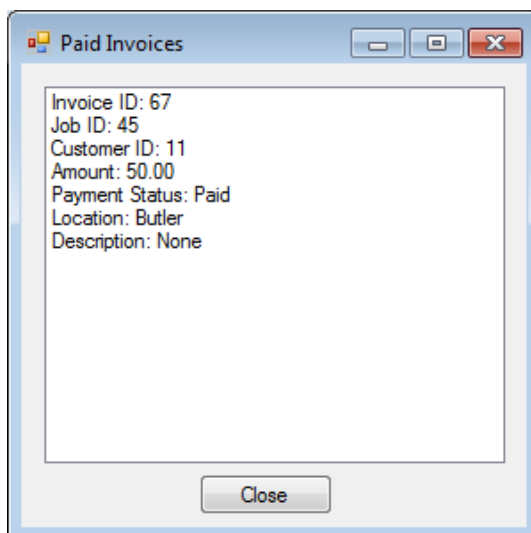
This is where a user searches for an invoice to delete



A dialog box titled "Delete Invoice" with a standard Windows-style title bar (minimize, maximize, close buttons). The main area contains a label "Search by ID:" followed by a text input field. Below the input field are two buttons: "Search" and "Cancel".

Display the Paid Invoices Page

This is where a user views the paid invoices



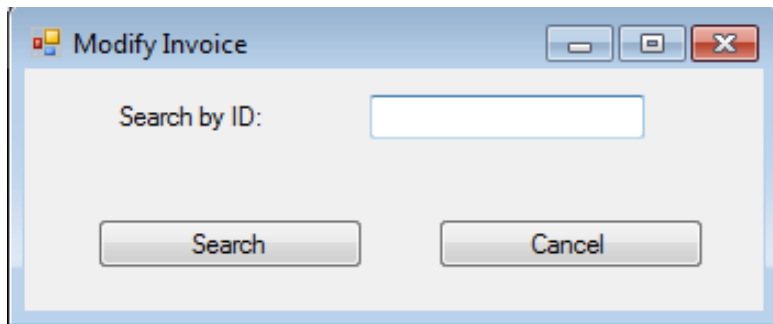
A dialog box titled "Paid Invoices" with a standard Windows-style title bar. The main area contains a list of invoice details:

- Invoice ID: 67
- Job ID: 45
- Customer ID: 11
- Amount: 50.00
- Payment Status: Paid
- Location: Butler
- Description: None

At the bottom of the dialog box is a "Close" button.

Modify an Invoice Search Page

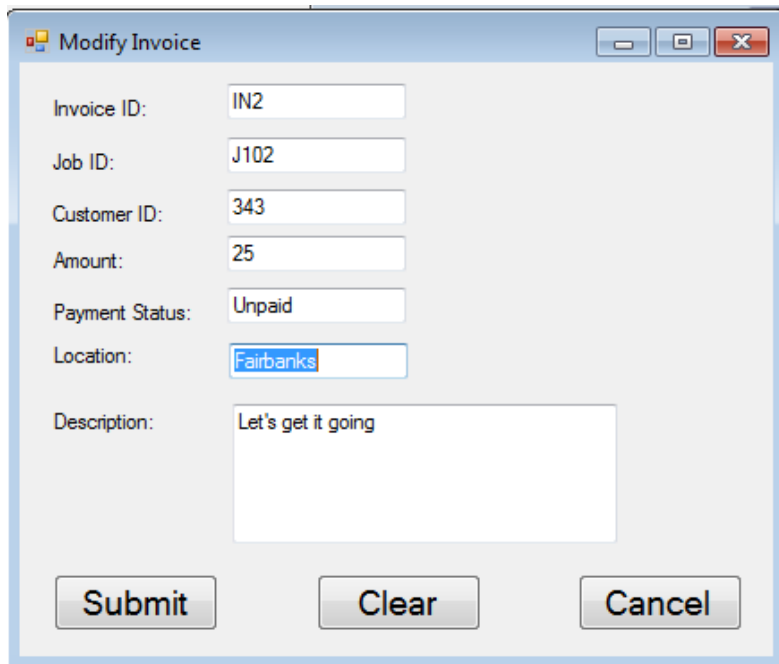
This is where a user searches for an invoice to modify



A screenshot of a Windows-style dialog box titled "Modify Invoice". It features a search bar labeled "Search by ID:" with an empty text input field. Below the search bar are two buttons: "Search" and "Cancel". The dialog box has standard Windows window controls (minimize, maximize, close) in the top right corner.

Modify an Invoice Page

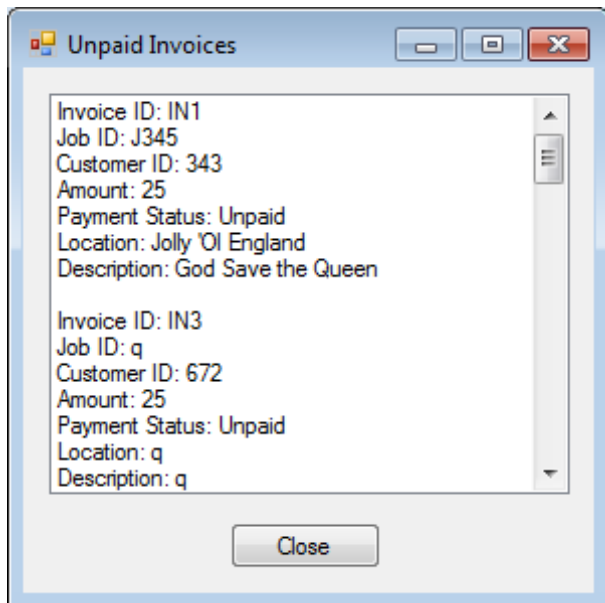
This is where a user modifies an invoice



A screenshot of a Windows-style dialog box titled "Modify Invoice". It contains several labeled text input fields: "Invoice ID:" with the value "IN2", "Job ID:" with "J102", "Customer ID:" with "343", "Amount:" with "25", "Payment Status:" with "Unpaid", and "Location:" with "Fairbanks". Below these is a larger text area for "Description:" containing the text "Let's get it going". At the bottom of the dialog are three buttons: "Submit", "Clear", and "Cancel". The dialog box has standard Windows window controls in the top right corner.

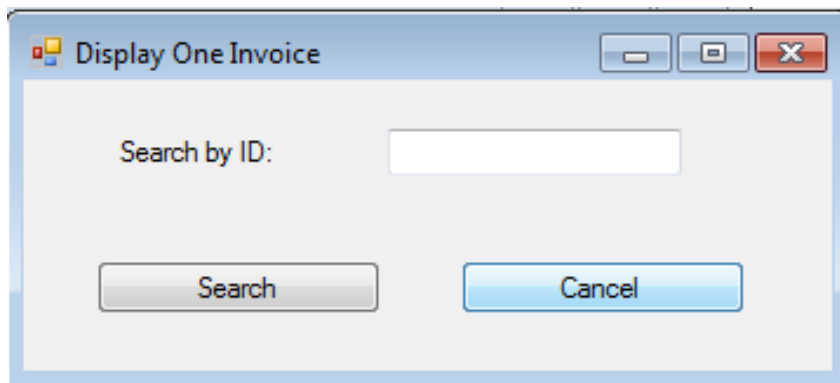
Unpaid Invoices Page

This is where a user views the unpaid invoices



Display an Invoice Search Page

This is where a user searches for a specific invoice to display

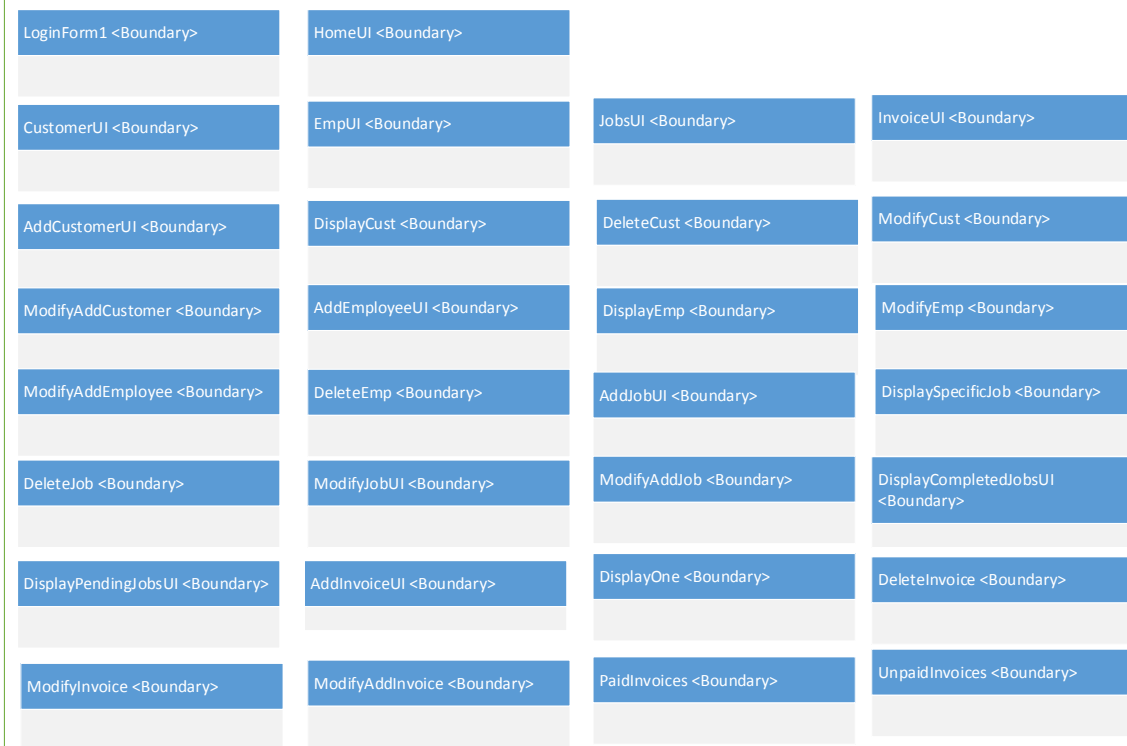


Package Diagrams for Design Organization

Our package diagram for Iteration 2 contains four different folders, to organize the types of classes that we implemented in the code. The layering of packages maps 100% to our class diagram shown above. The first folder contains all of our boundary classes, as they are the forms and GUIs the users are interacting with on the front-end. In the middle are our Controller and System folders. These middle packages are where all of the logic takes place, whether it deals with the lists of entities or the displaying and saving of files. The bottom package contains all of our entity classes as they are the back-end classes, those that are written to and read from the text files. Please see our diagram on the next page.

Just the Job

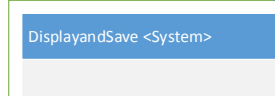
Boundaries



Controllers



System



Entities



Implementation

Tested Code

We do not have our tested code from this iteration because we used the same Visual Basic file, overwriting the old code with our updated code. It is important to note that in this iteration, we did implement entity classes. Our code took huge strides in this iteration, as we added new forms to accommodate for the new functionality, consolidated our controllers, and created entities to hold our attribute information. In general, our code was modeled in parallel with our design package diagram above.

Package Diagrams for Implementation Organization

Because we do not have our tested code from this iteration, we do not have a package diagram for our implemented code. At this point in Iteration 2 our design package diagram was mapped 100% to our code, so please refer to our design package diagram above.

Chapter 3: Project Iteration 3

Brief Description of Work

In Iteration 3 we were asked to design and implement requirements 5a-e, 6a-c, and 7a-d.

Requirement 5 is managing customer invoices for regular jobs, which means we had to modify the job code slightly to accommodate for one-time and regular jobs. Requirement 6 is managing customer payments, so we had to create new classes that allowed for the addition of payments.

These payments directly interact with invoices because once an invoice has received enough payments, it is marked as paid. Requirement 7 is maintaining the manager's personal weekly schedule, so we had to implement that functionality through another entity class. The addition of these classes was an easy transition to make, as we kept the same architecture of classes from past iterations. The final requirement we handled in this iteration was differentiating between manager and receptionist logins. In the first two iterations the receptionist could make deletions throughout the system, and we wanted to disallow this.

Static Design Model

Final Class Diagram

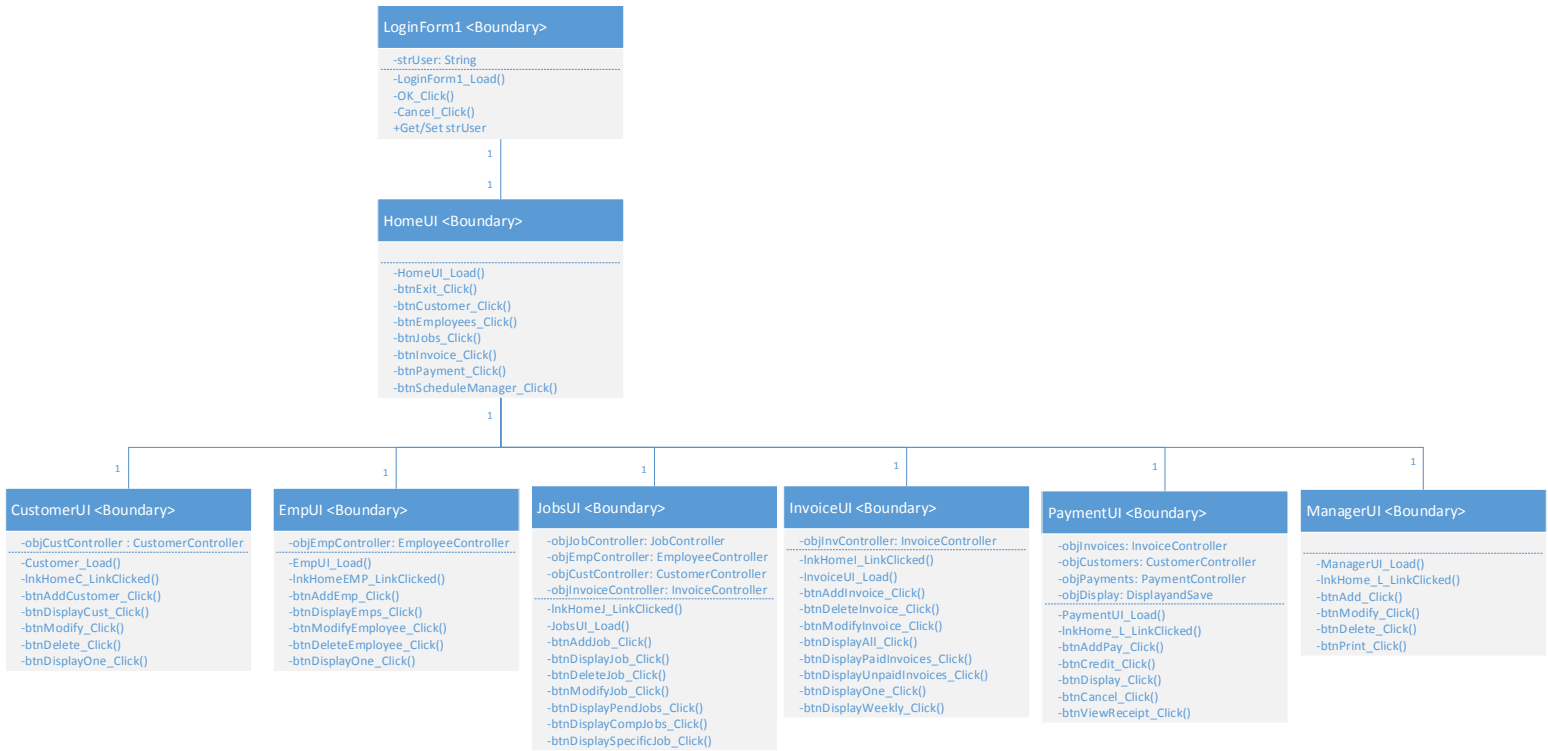
Once we reached Iteration 3, our final class diagram took shape and we had a structure we were pleased with. Only a few things changed in our class diagram from Iteration 2 to Iteration 3, one being the addition of start and end date attributes to our Job class, and also the addition of support classes for our new two entity classes, Payment and Appointment. There were also slight modifications made in the System class, to accommodate for our new entities. To see our full class diagram with all associations, please see our Wiki.

System Class

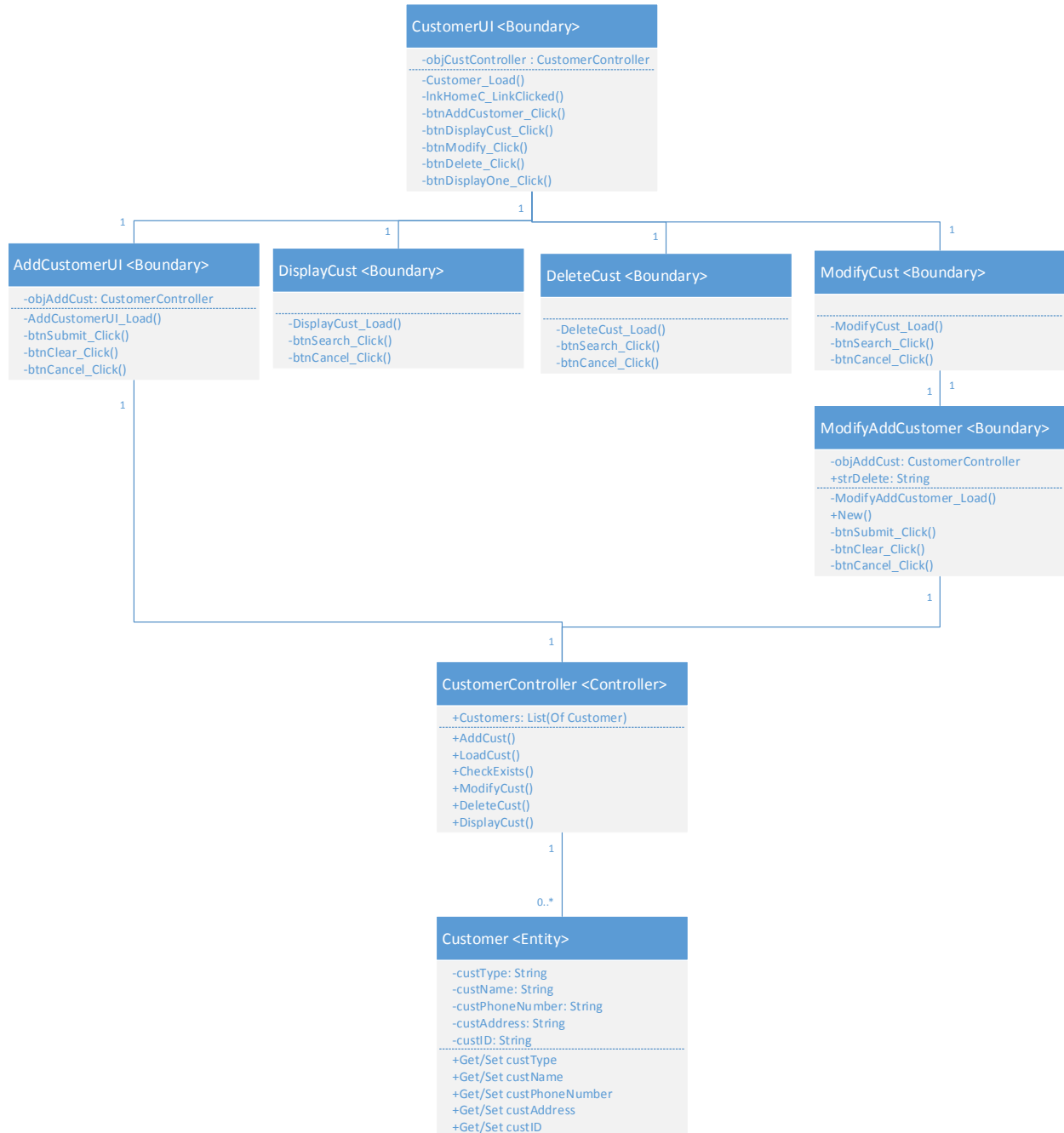
DisplayandSave <System>

```
+SubmitCust()  
+DisplayCust()  
+LoadCust()  
+deleteCust()  
+SubmitEmp()  
+DisplayEmp()  
+LoadEmp()  
+deleteEmp()  
+SubmitJob()  
+DisplayJob()  
+LoadJob()  
+DeleteJob()  
+SubmitInvoice()  
+DisplayInvoice()  
+LoadInvoice()  
+DeleteInvoice()  
+SubmitPayment()  
+LoadPayments()  
+DisplayPayments()  
+DeletePayment()  
+SubmitAppointment()  
+LoadAppointments()  
+DeleteAppointment()
```

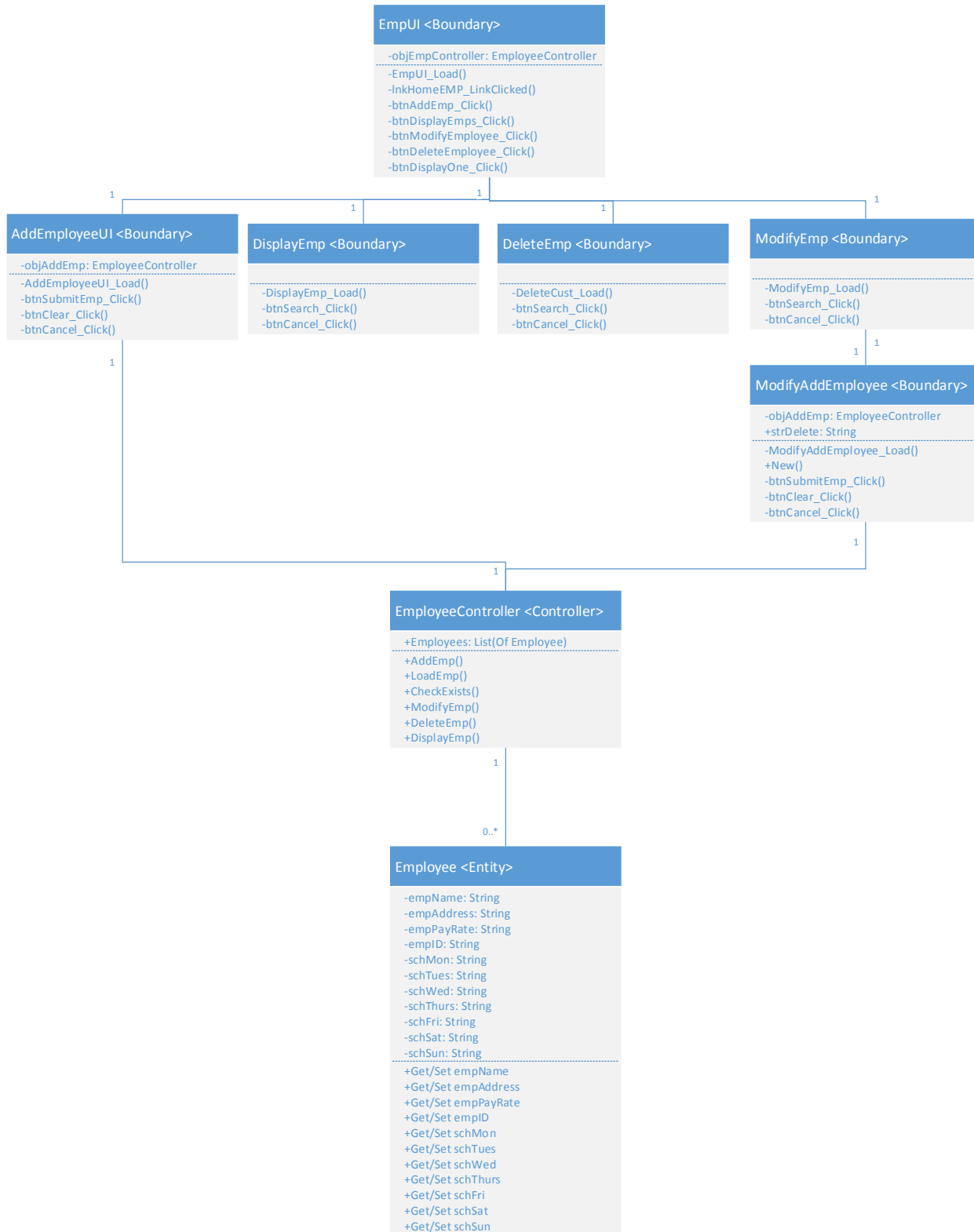
Outer Boundaries



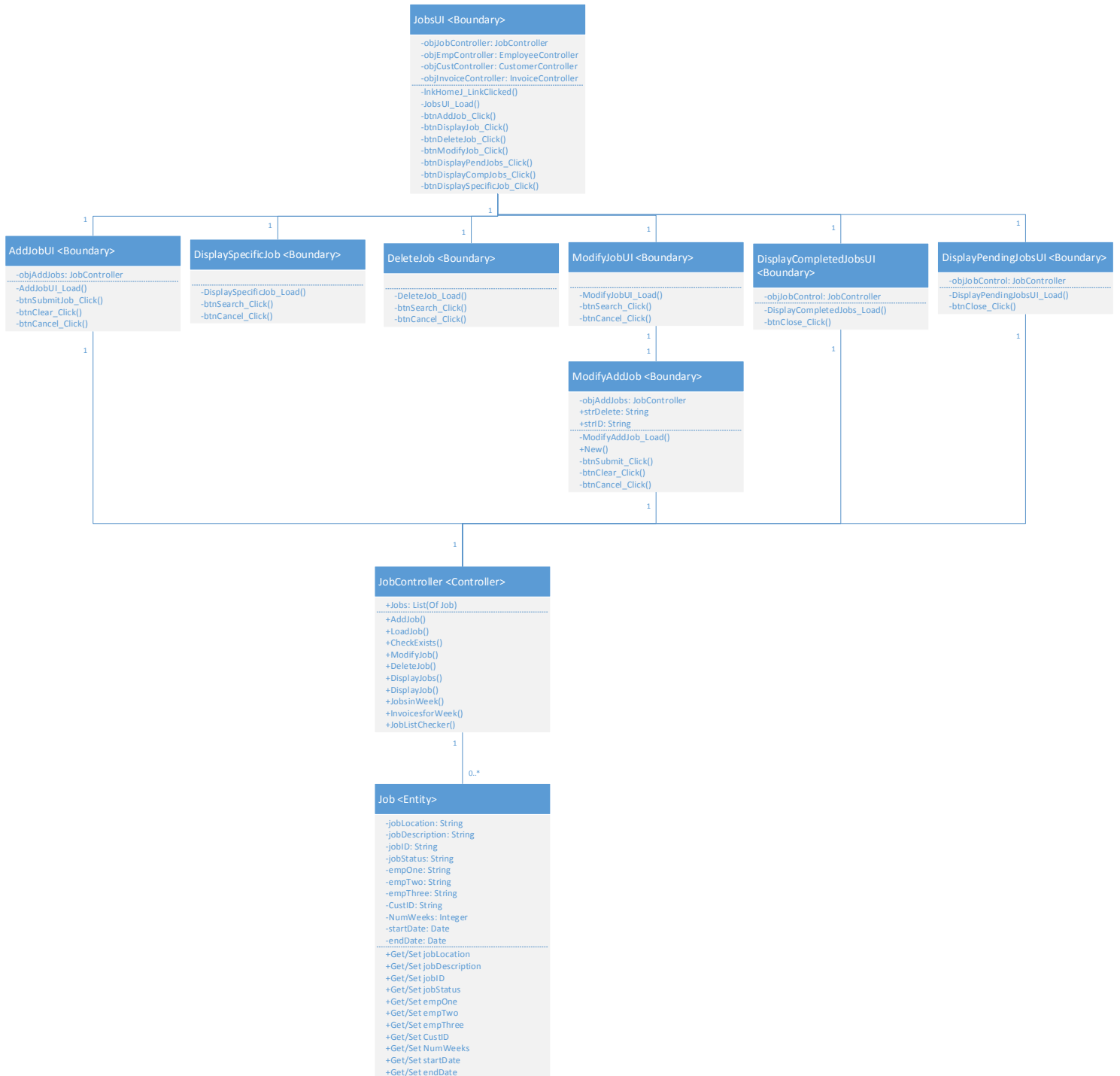
Customer Classes



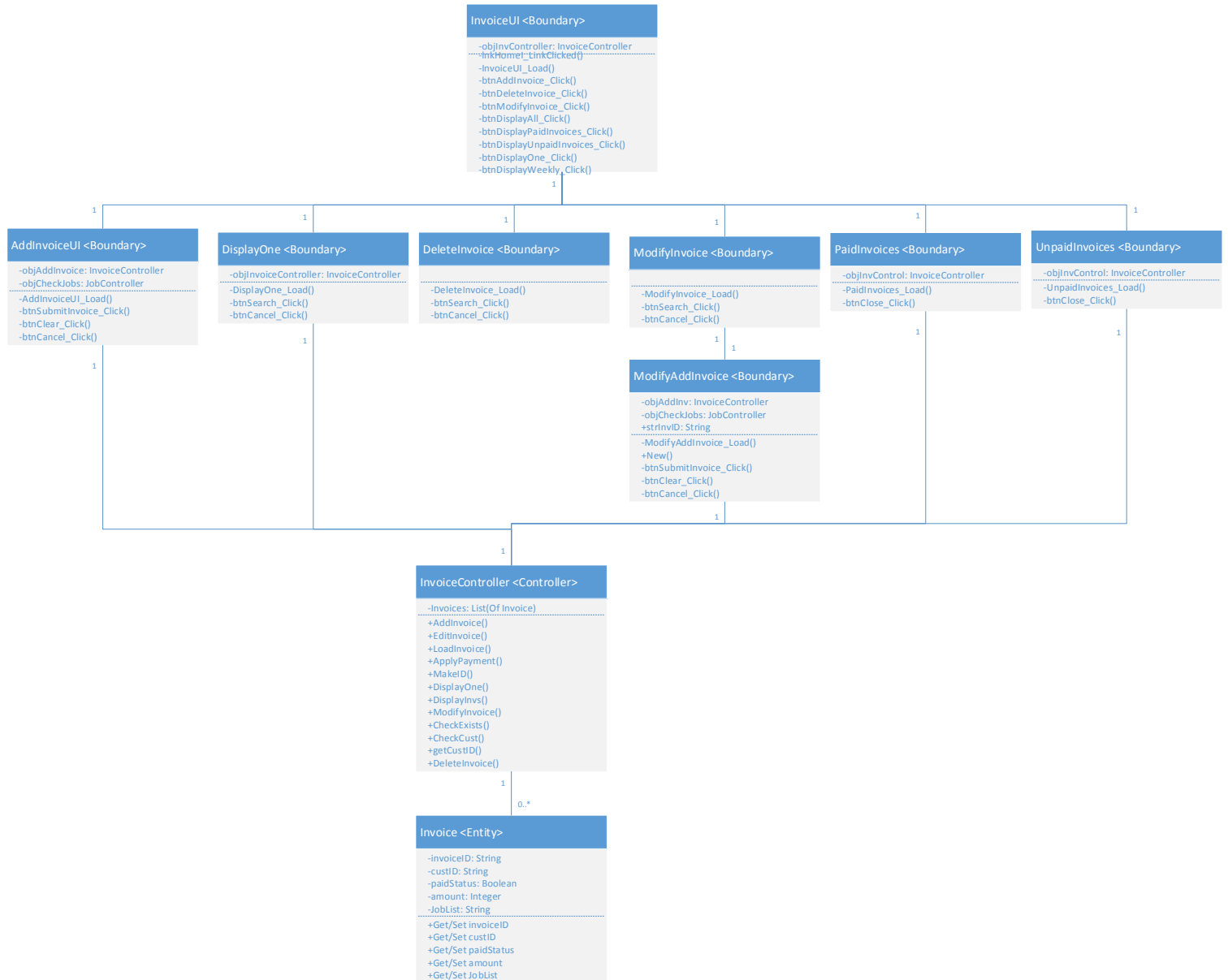
Employee Classes



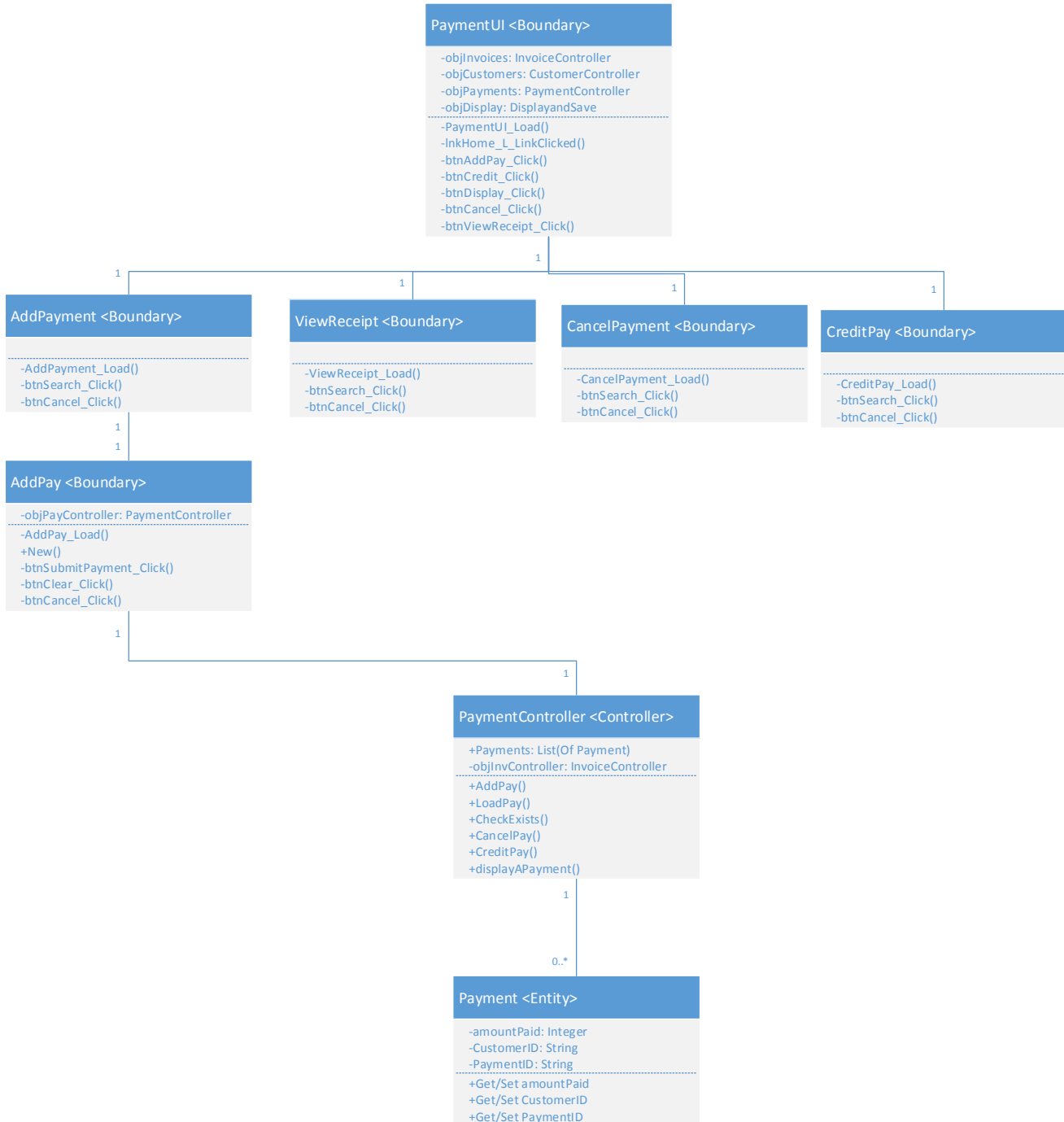
Jobs Classes



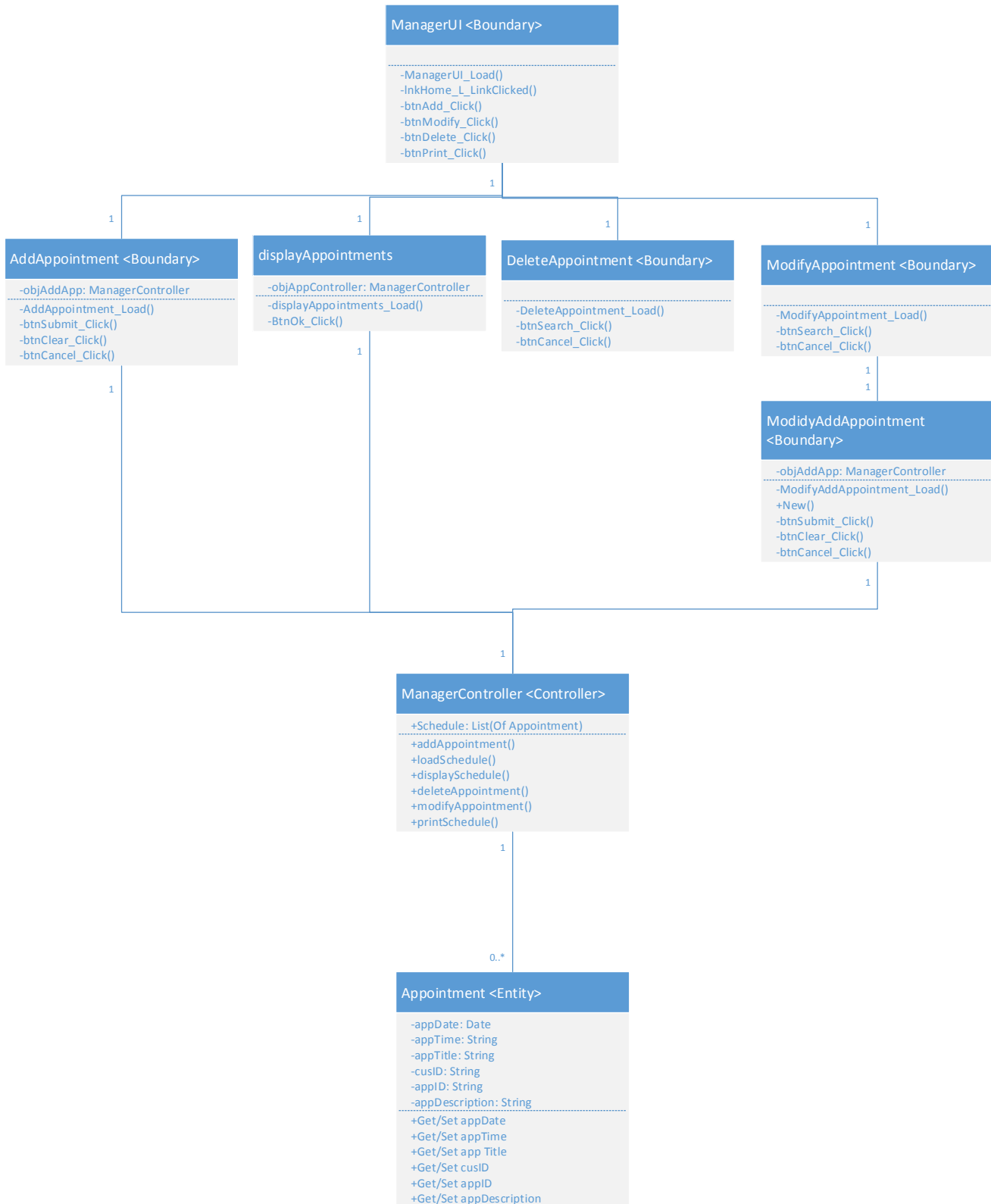
Invoice Classes



Payment Classes

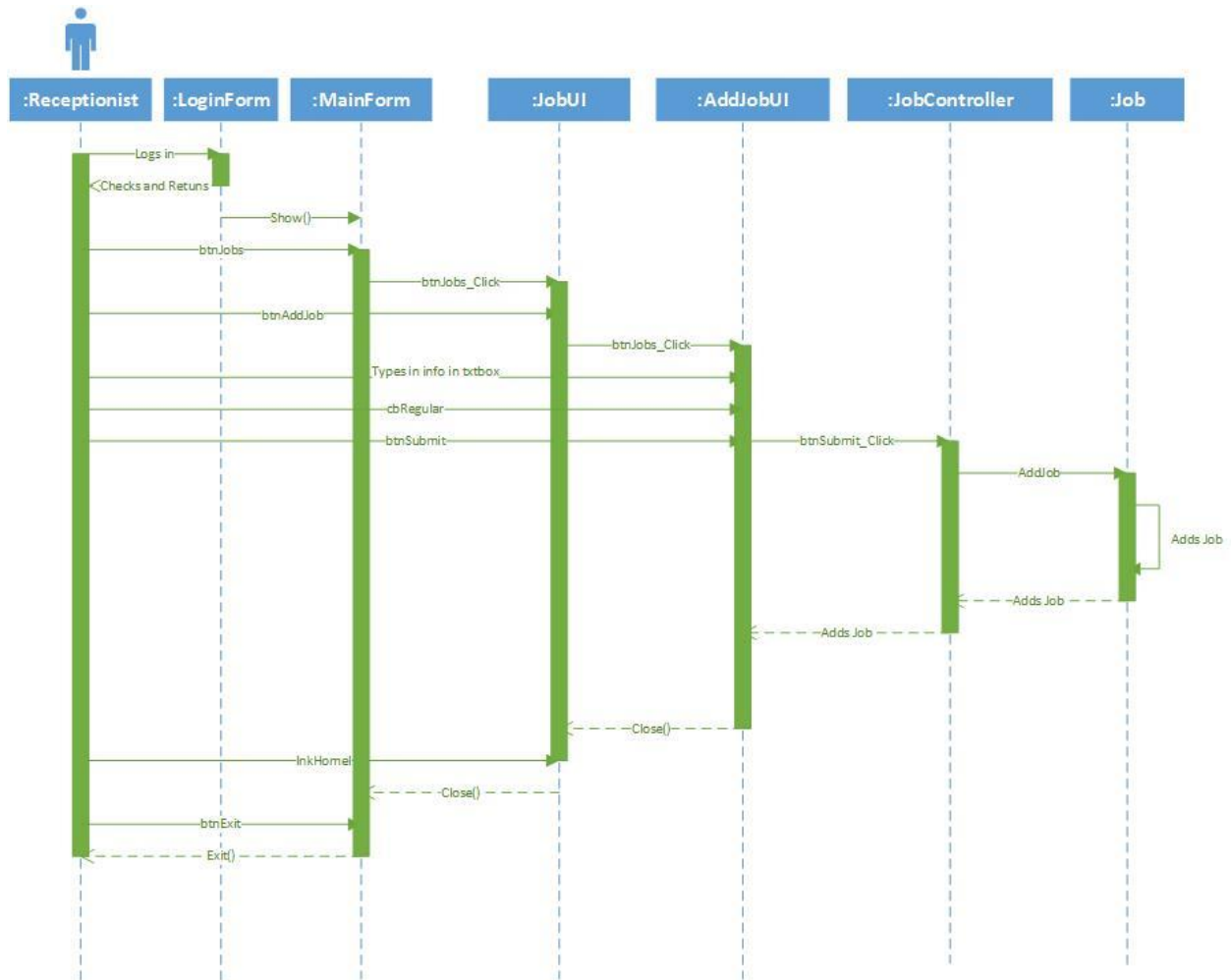


Manager Classes

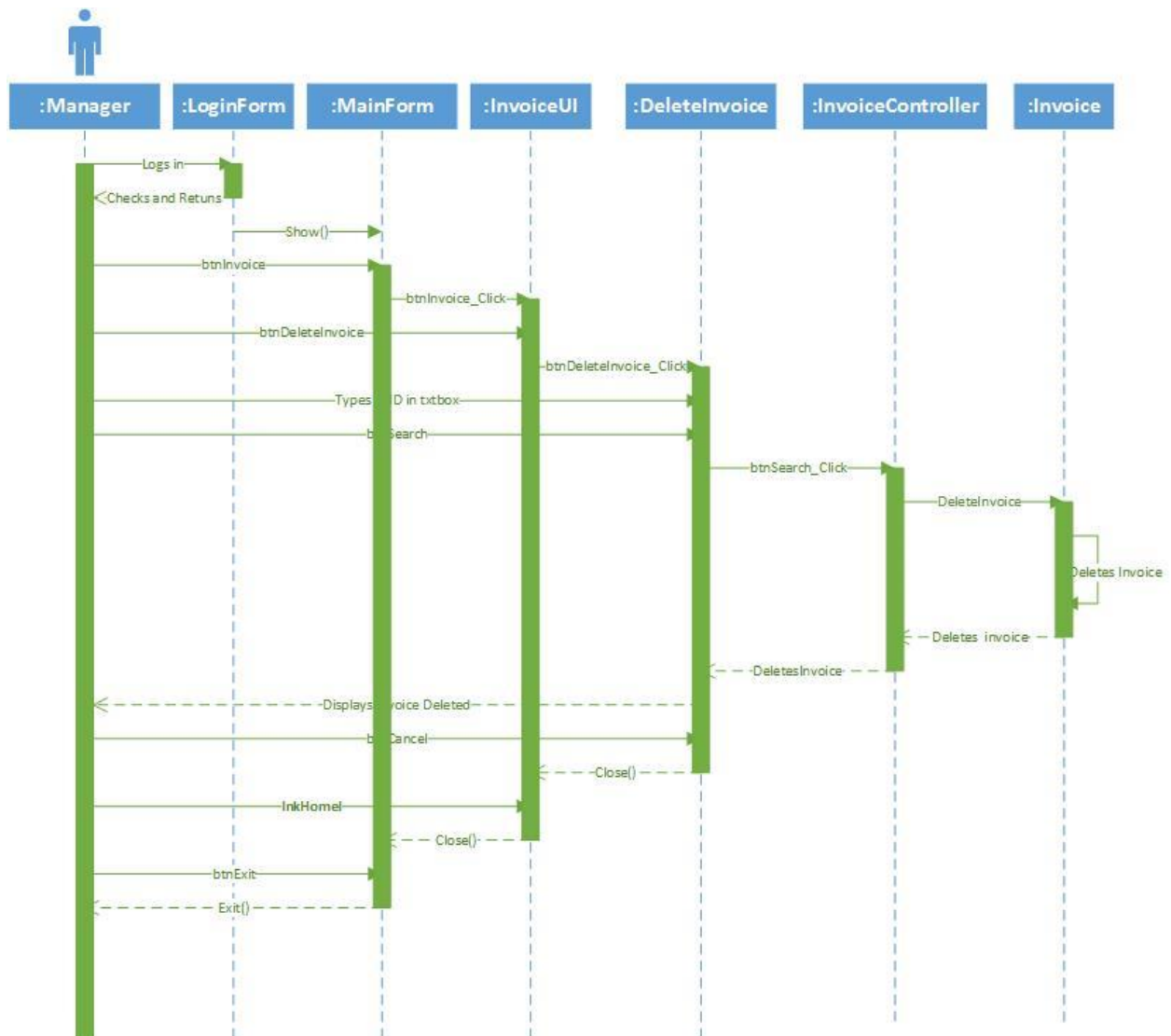


Dynamic Design Model

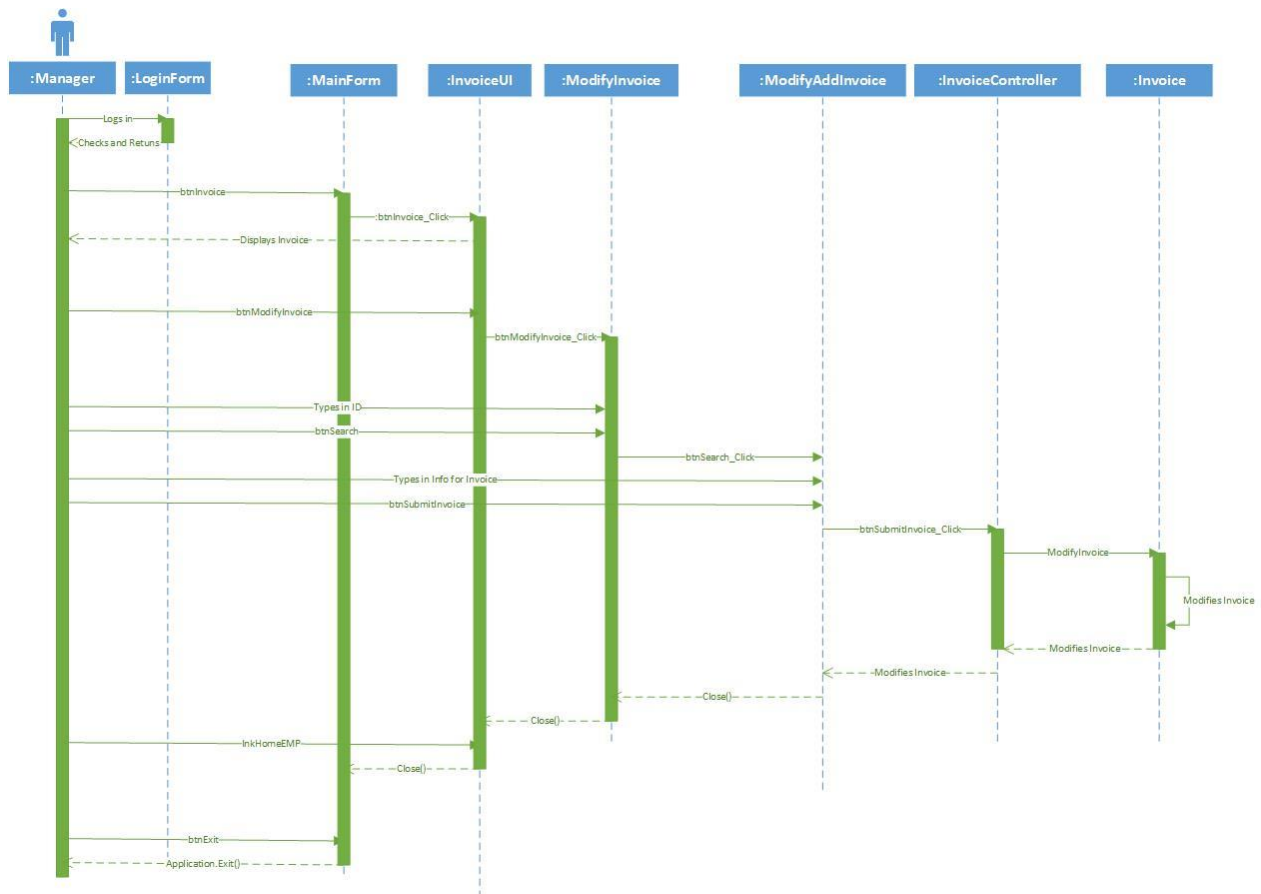
This is a detailed sequence diagram for 5a) Create a new invoice



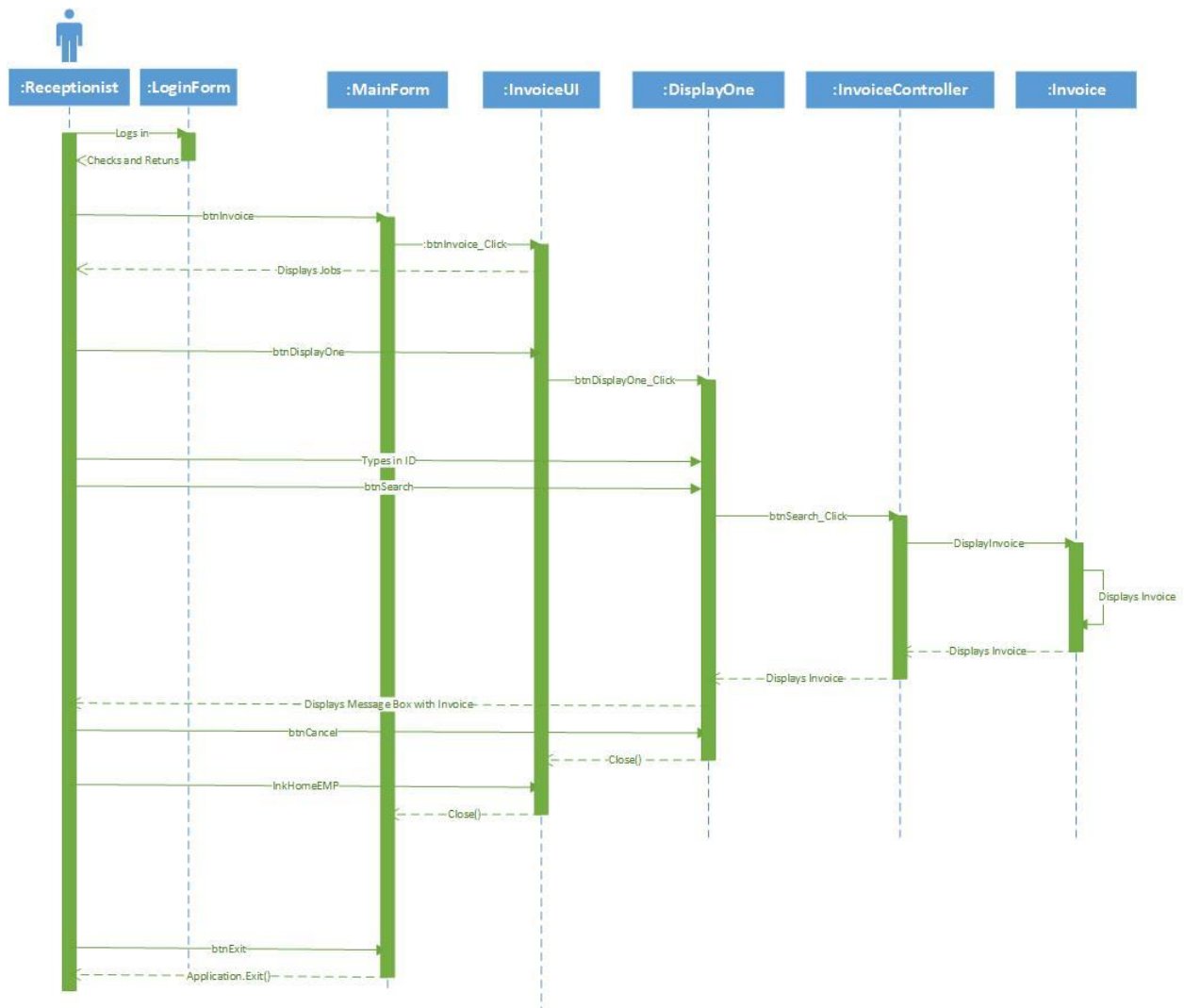
This is a detailed sequence diagram for 5b) Cancel an existing invoice



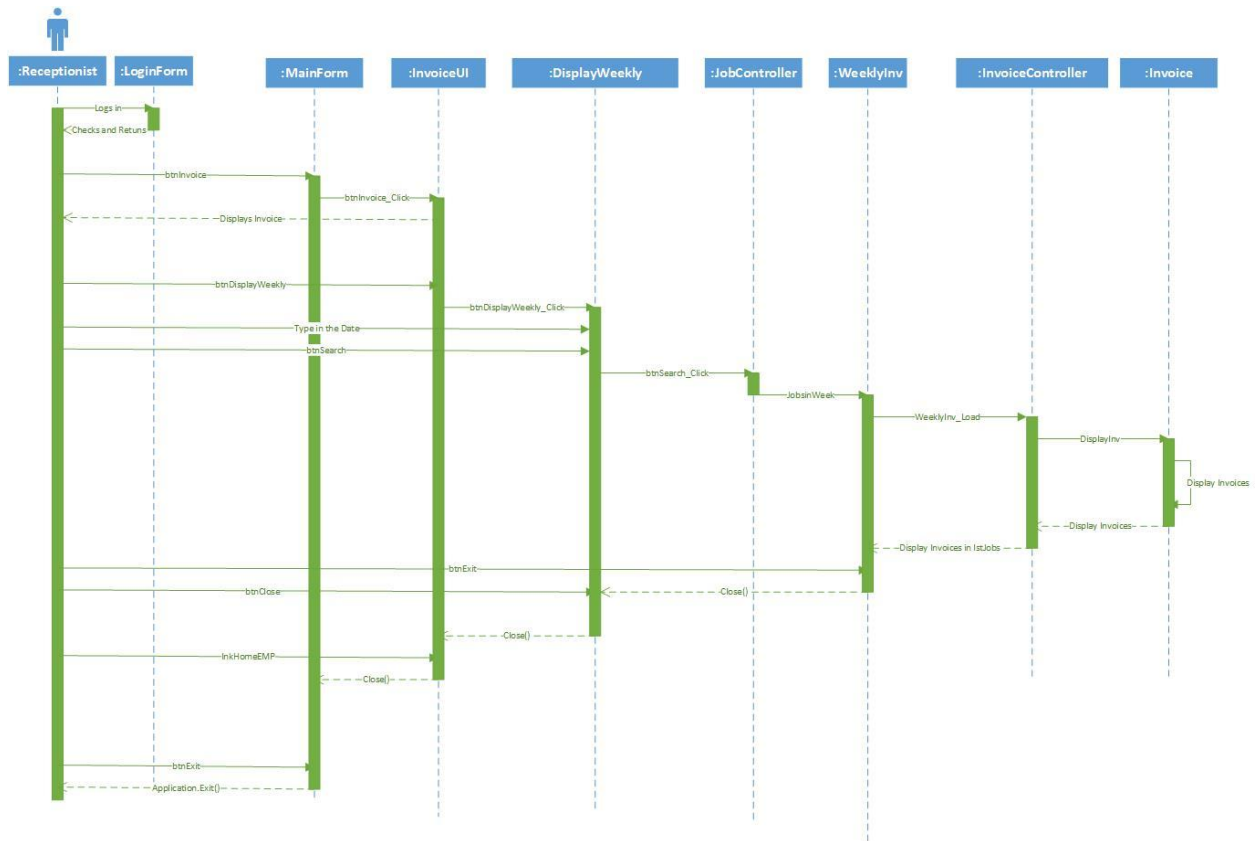
This is a detailed sequence diagram for 5c) Modify an existing invoice



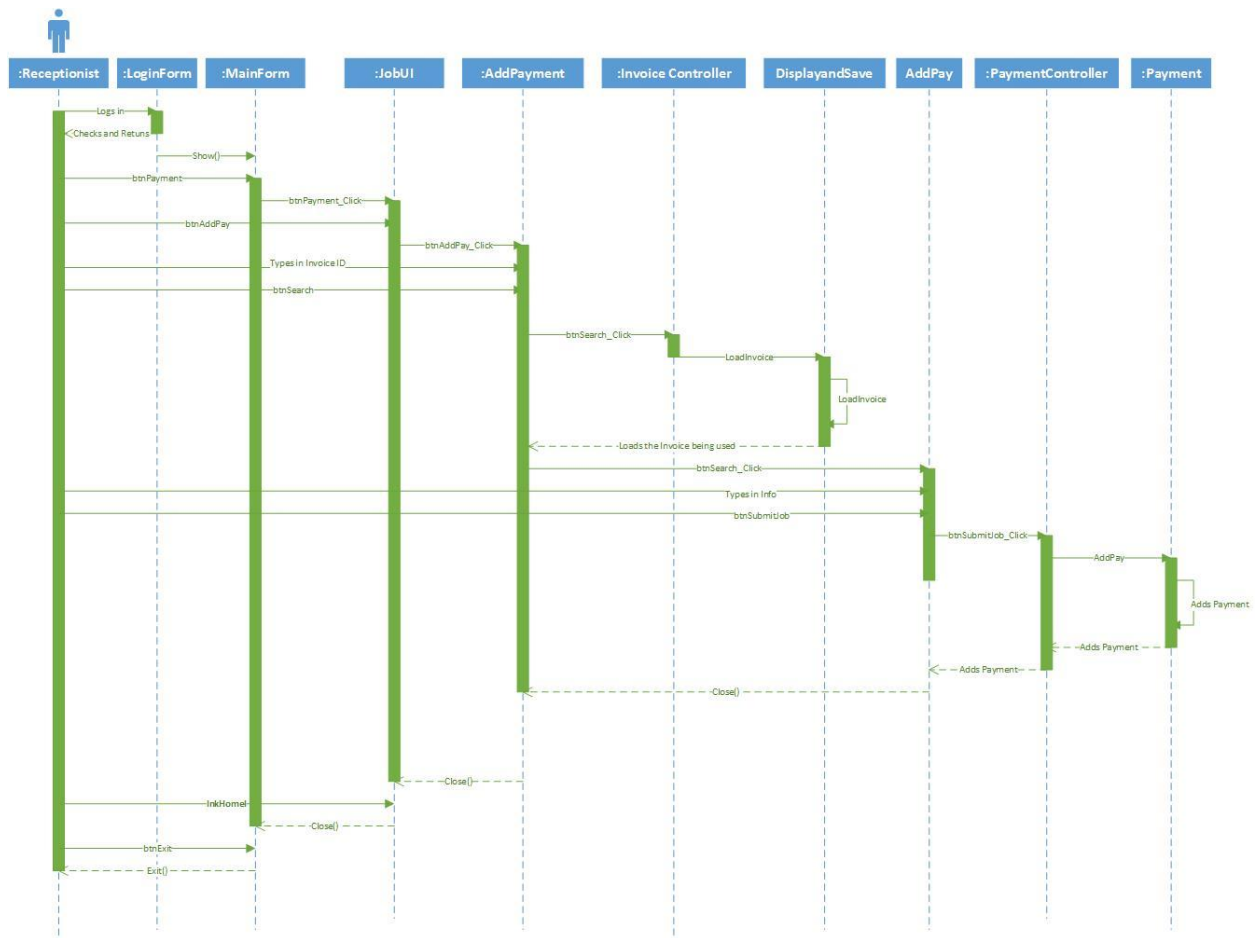
This is a detailed sequence diagram for 5d) Display/print an existing invoice



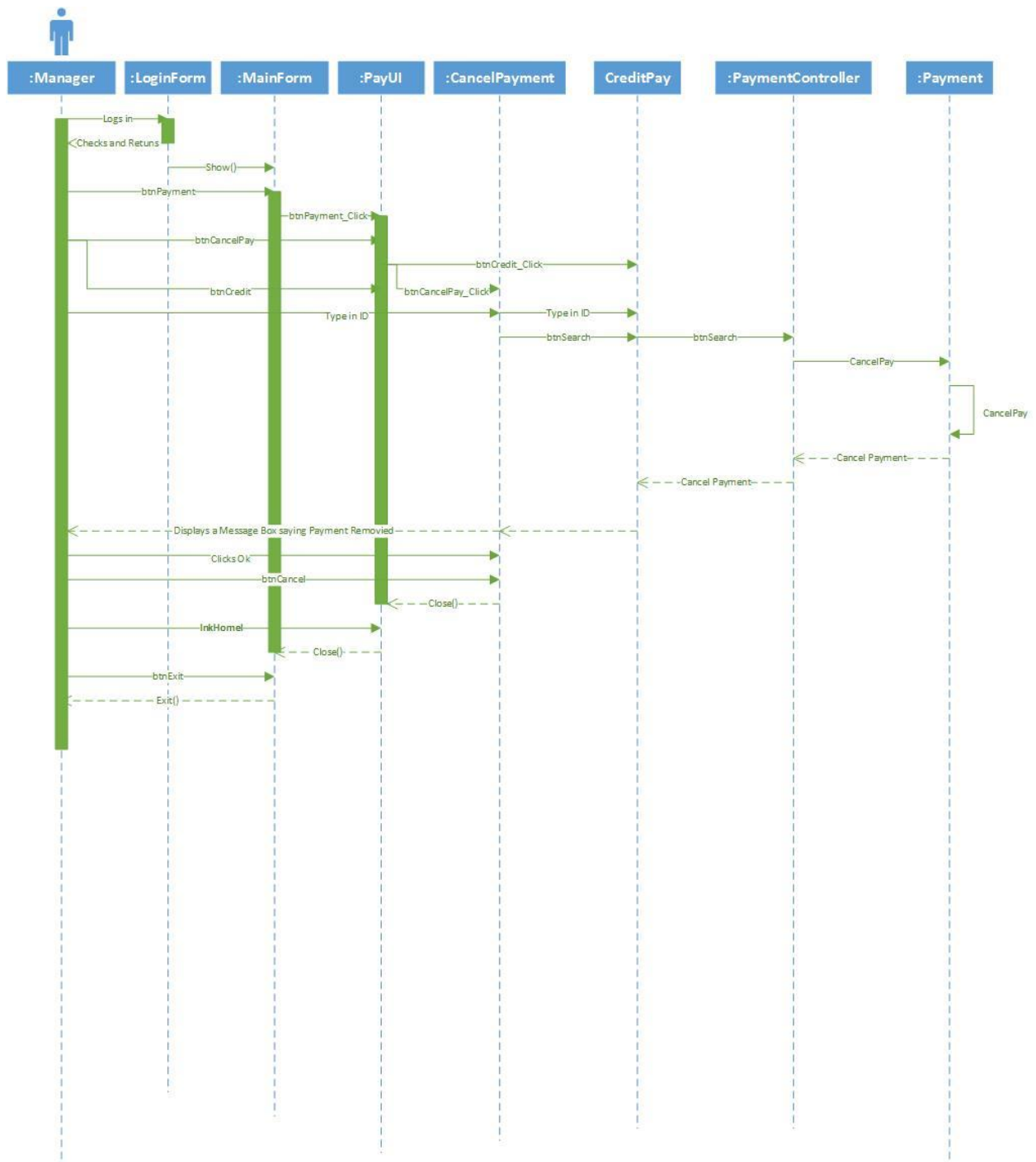
This is a detailed sequence diagram for 5e) Display/print a list of all invoices for a given week with a payment status



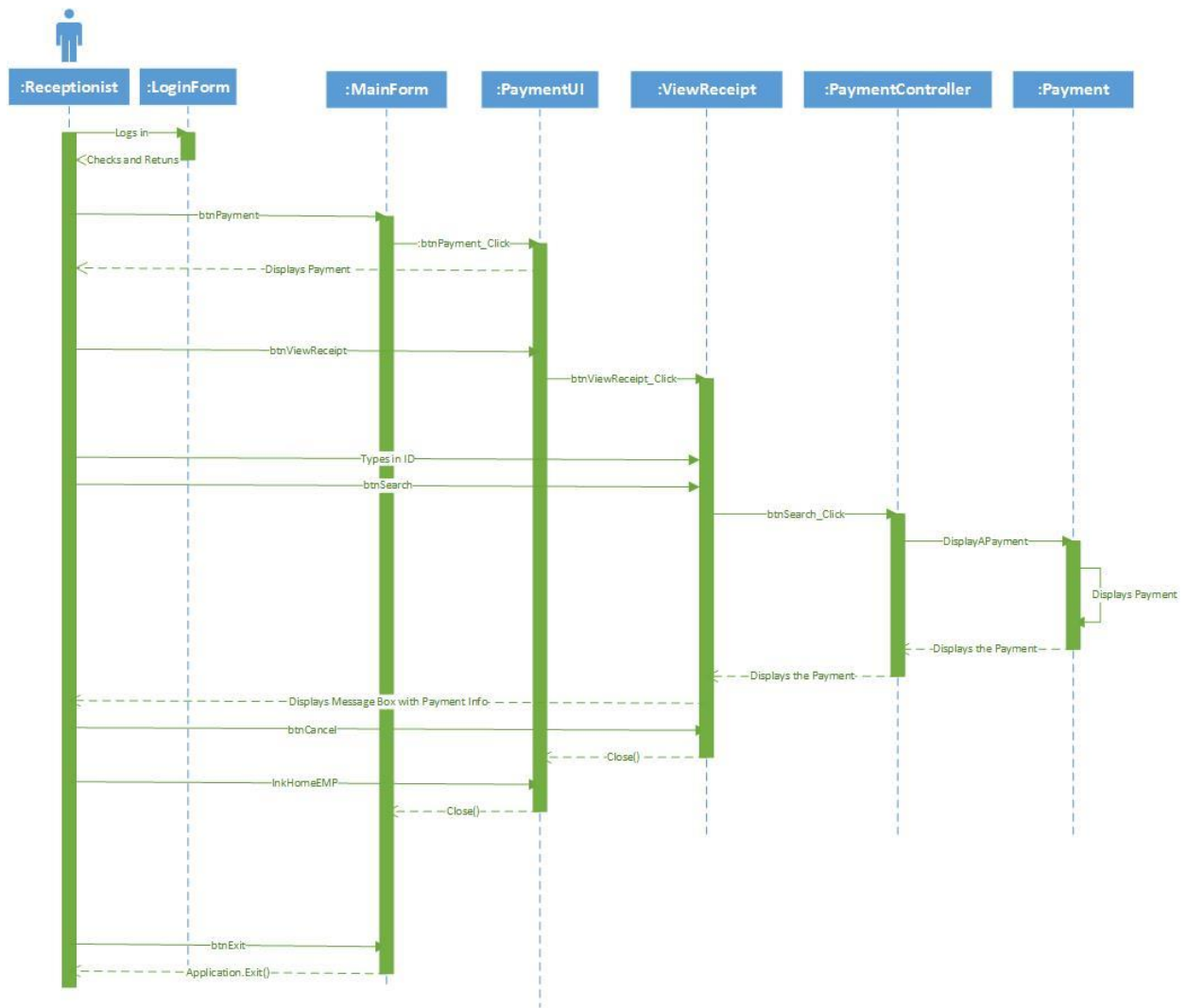
This is a detailed sequence diagram for 6a) Record a full payment



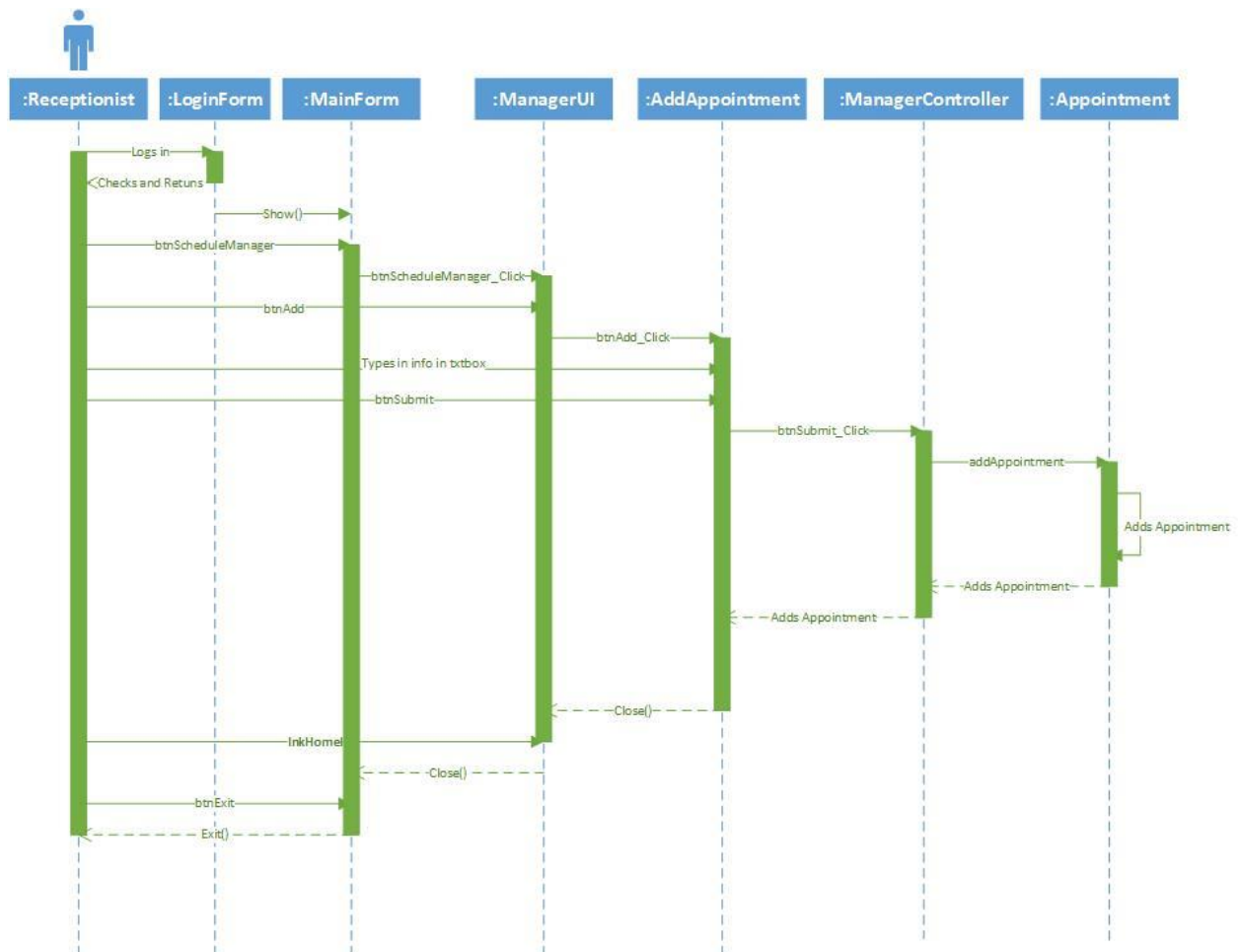
This is a detailed sequence diagram for 6b) Cancel/credit a payment



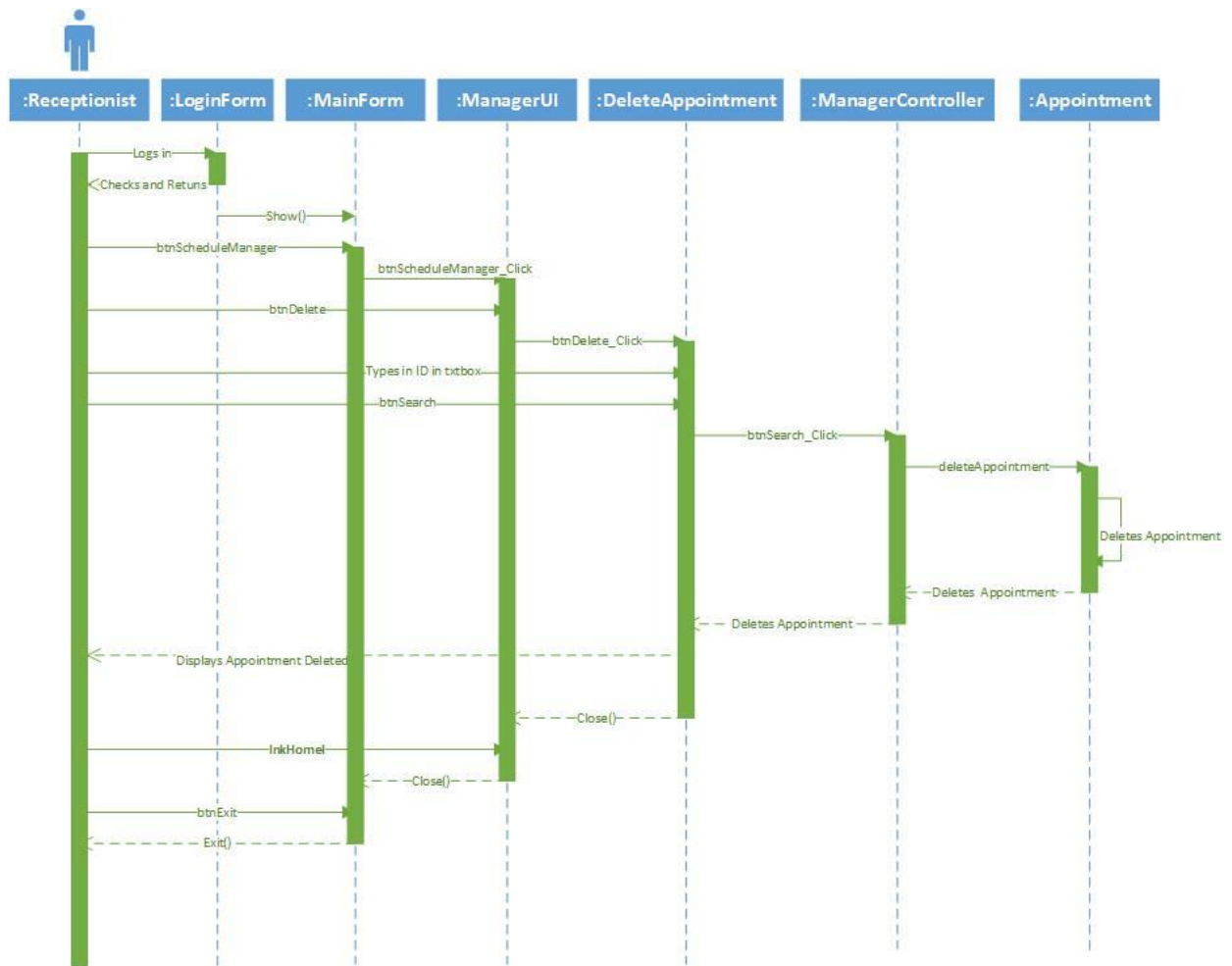
This is a detailed sequence diagram for 6c) Print a receipt



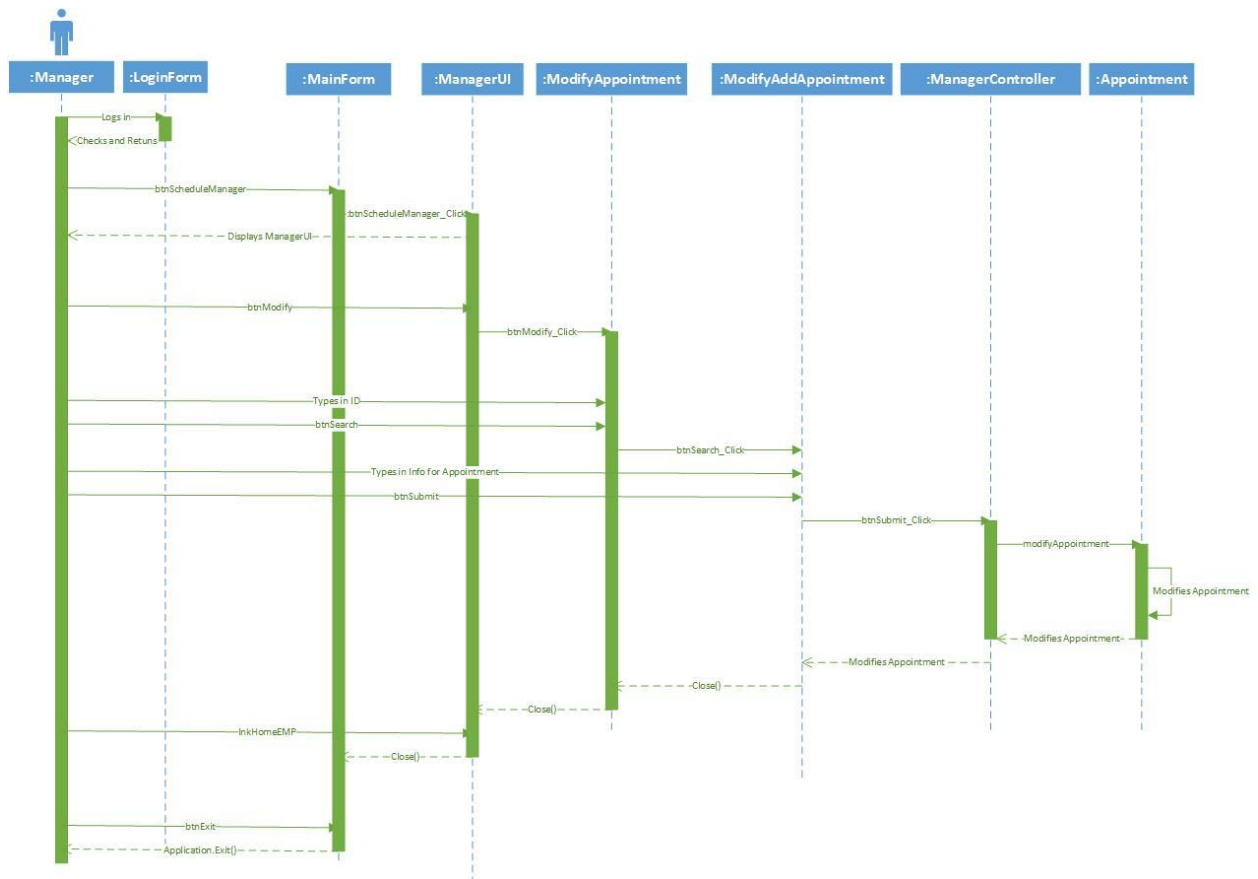
This is a detailed sequence diagram for 7a) Add a new appointment



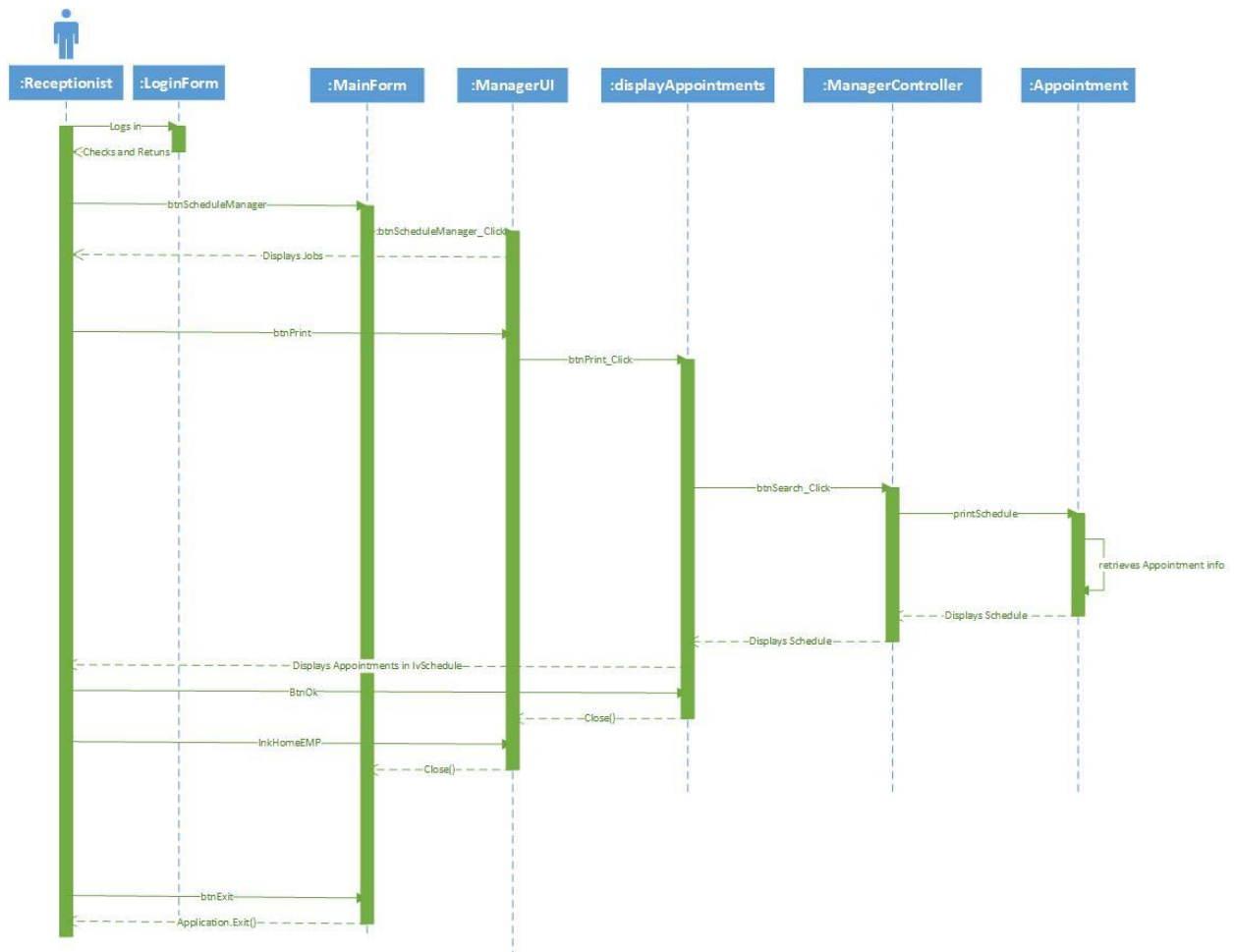
This is a detailed sequence diagram for 7b) Cancel an existing appointment



This is a detailed sequence diagram for 7c) Modify an existing appointment



This is a detailed sequence diagram for 7d) Print weekly schedule



GUI Model

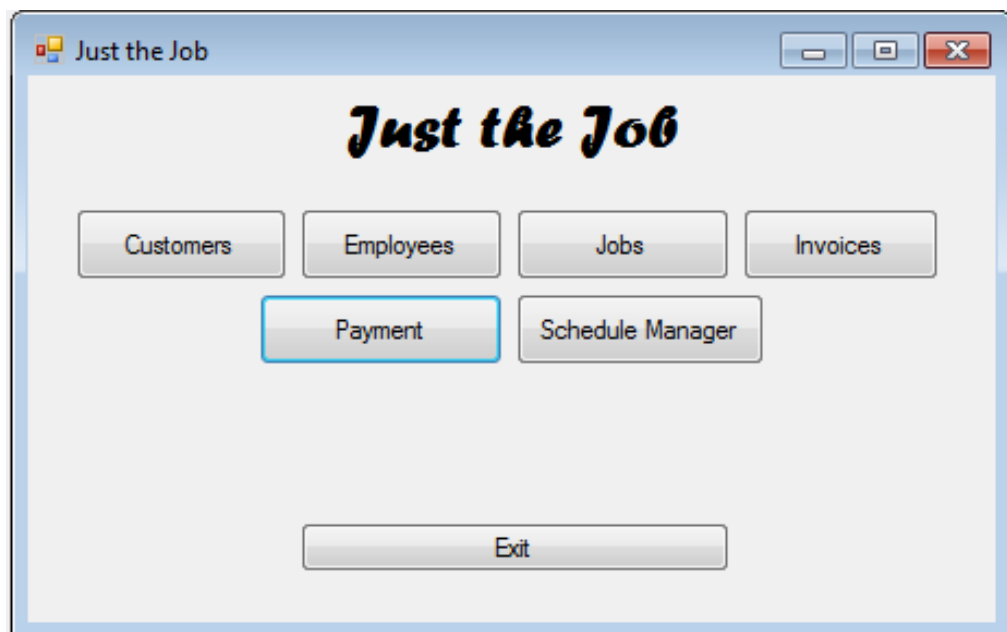
Login Page

This is where a user logs in



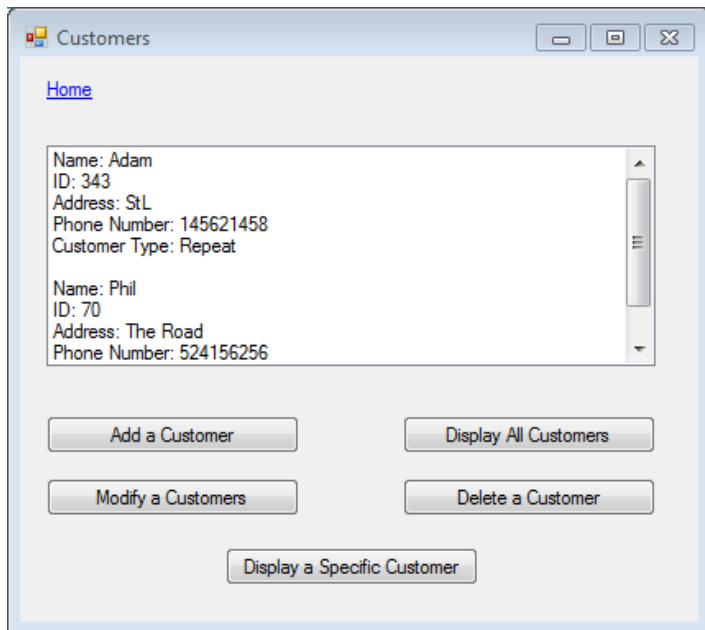
Home Page

This is where a user navigates to the various categories



Customers Page

This is where a user sees all the customers and performs various requirements to customers



The screenshot shows a window titled "Customers" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a blue hyperlink labeled "Home" at the top left. Below it is a scrollable text area containing two customer records. The first record is for Adam (ID: 343, Address: StL, Phone Number: 145621458, Customer Type: Repeat). The second record is for Phil (ID: 70, Address: The Road, Phone Number: 524156256). At the bottom of the window, there are five buttons: "Add a Customer", "Display All Customers", "Modify a Customers", "Delete a Customer", and "Display a Specific Customer".

Customers

[Home](#)

Name: Adam
ID: 343
Address: StL
Phone Number: 145621458
Customer Type: Repeat

Name: Phil
ID: 70
Address: The Road
Phone Number: 524156256

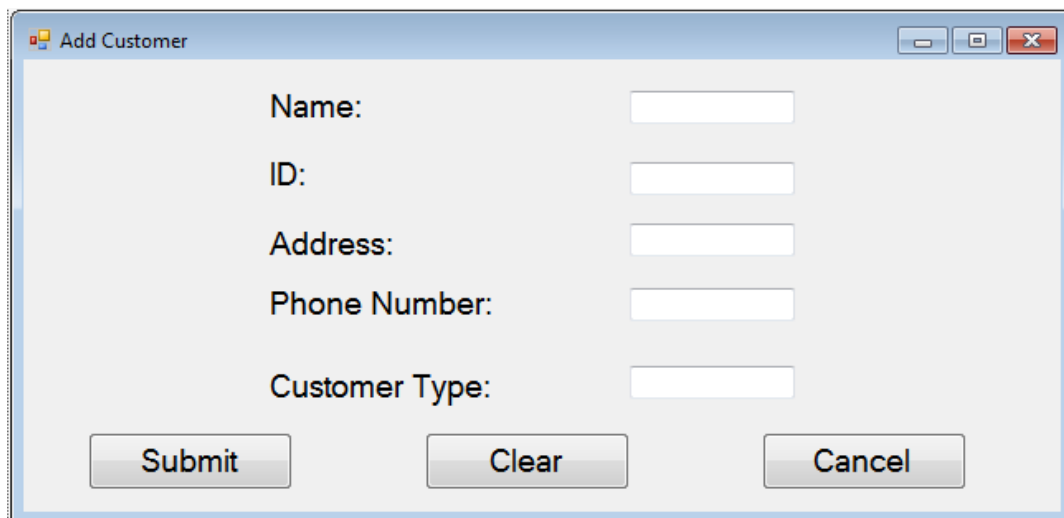
Add a Customer Display All Customers

Modify a Customers Delete a Customer

Display a Specific Customer

Adds Customer Page

This is the form to add a customer



The screenshot shows a window titled "Add Customer" with a standard Windows-style title bar (minimize, maximize, close buttons). The form contains five labeled input fields: "Name:", "ID:", "Address:", "Phone Number:", and "Customer Type:". Each label is followed by a text input box. At the bottom of the window, there are three buttons: "Submit", "Clear", and "Cancel".

Add Customer

Name:

ID:

Address:

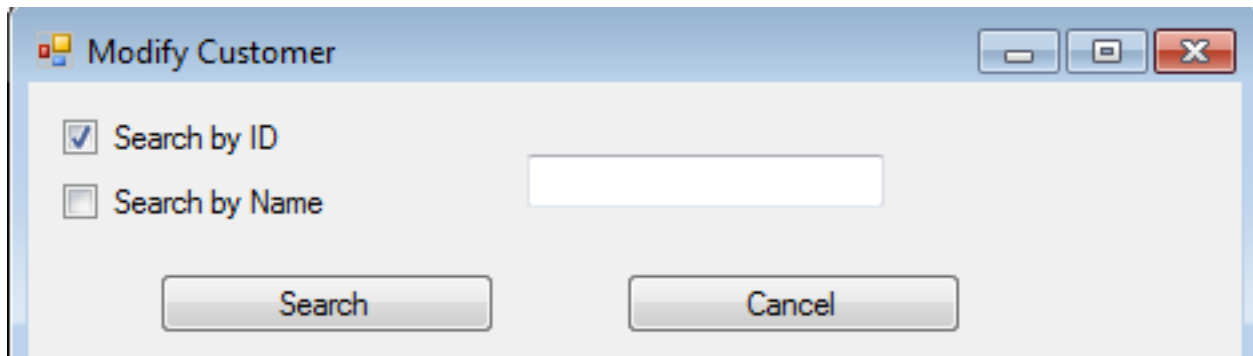
Phone Number:

Customer Type:

Submit Clear Cancel

Modify Customer Search Page

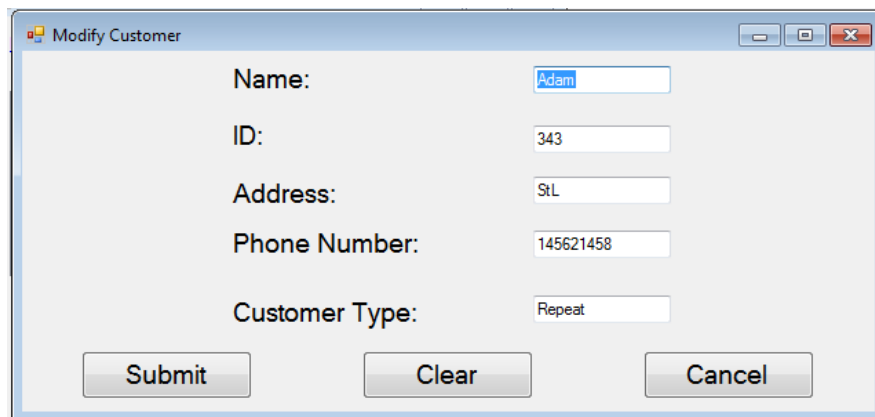
This is where a user searches for the customer he wants to modify



A screenshot of a web application window titled "Modify Customer". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a text input field. Below the input field are two buttons: "Search" and "Cancel".

Modify a Customer Page

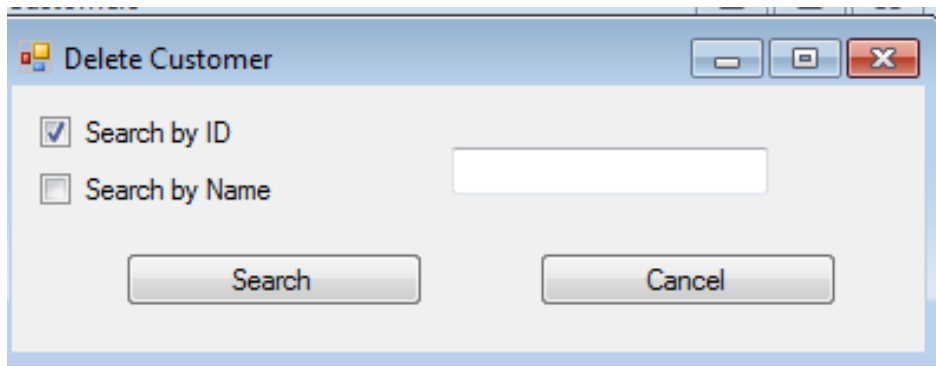
This is where the user modifies a customer's info



A screenshot of a web application window titled "Modify Customer". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are five labeled text input fields arranged vertically: "Name:" (containing "Adam"), "ID:" (containing "343"), "Address:" (containing "StL"), "Phone Number:" (containing "145621458"), and "Customer Type:" (containing "Repeat"). Below these fields are three buttons: "Submit", "Clear", and "Cancel".

Delete Customer Search Page

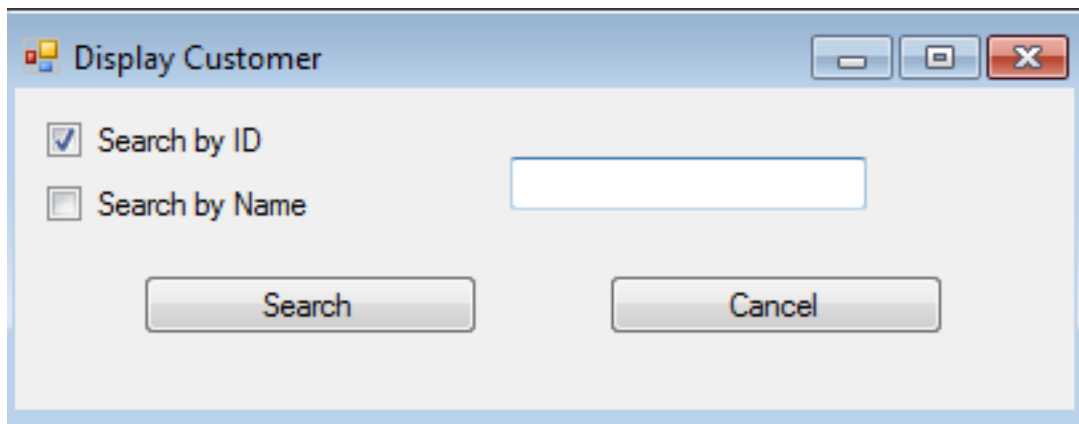
This is where the user searches for the customer they want to delete



A screenshot of a Windows-style dialog box titled "Delete Customer". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a text input field. At the bottom of the dialog are two buttons: "Search" and "Cancel".

Display Customer Search Page

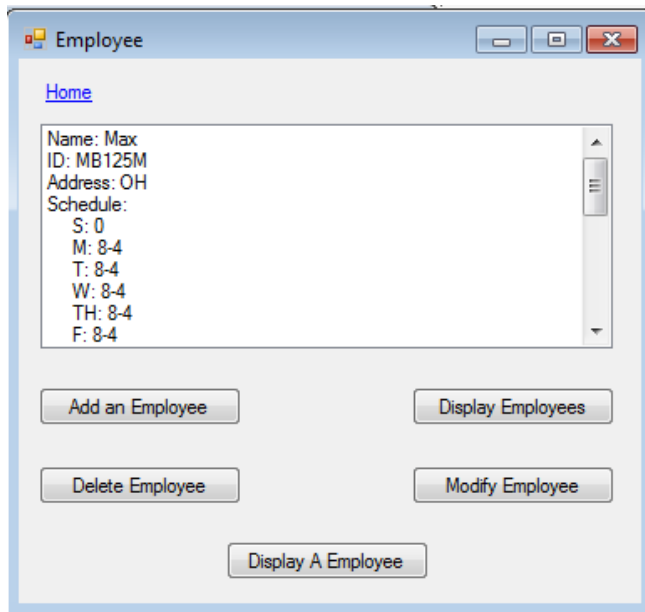
This is where a user searches for a specific customer to display



A screenshot of a Windows-style dialog box titled "Display Customer". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" (which is selected) and "Search by Name". To the right of these buttons is a text input field. At the bottom of the dialog are two buttons: "Search" and "Cancel".

Employee Page

This is where a user sees all the employees and performs various requirements to employees



The 'Employee' window displays a list of employee details in a text area. The details shown are: Name: Max, ID: MB125M, Address: OH, and Schedule: S: 0, M: 8-4, T: 8-4, W: 8-4, TH: 8-4, F: 8-4. Below the text area are five buttons: 'Add an Employee', 'Display Employees', 'Delete Employee', 'Modify Employee', and 'Display A Employee'.

Employee

[Home](#)

Name: Max
ID: MB125M
Address: OH
Schedule:
S: 0
M: 8-4
T: 8-4
W: 8-4
TH: 8-4
F: 8-4

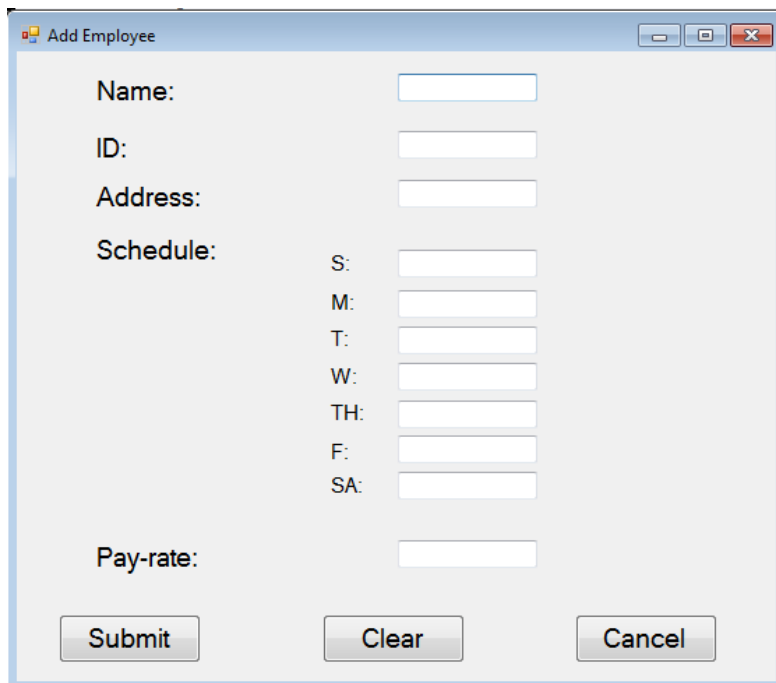
Add an Employee Display Employees

Delete Employee Modify Employee

Display A Employee

Add Employee Page

This is where a user adds an employee



The 'Add Employee' window contains input fields for employee information. The fields are: Name, ID, Address, and a group of seven fields for the schedule (S, M, T, W, TH, F, SA). There is also a field for Pay-rate. At the bottom are three buttons: 'Submit', 'Clear', and 'Cancel'.

Add Employee

Name:

ID:

Address:

Schedule:

S:

M:

T:

W:

TH:

F:

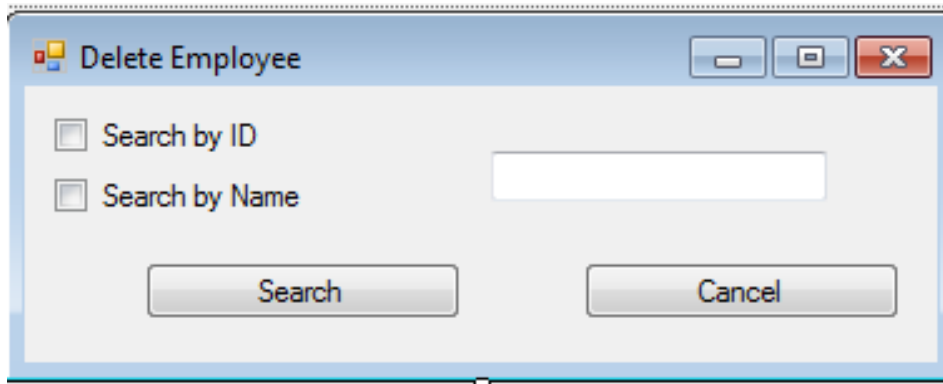
SA:

Pay-rate:

Submit Clear Cancel

Delete Employee Search Page

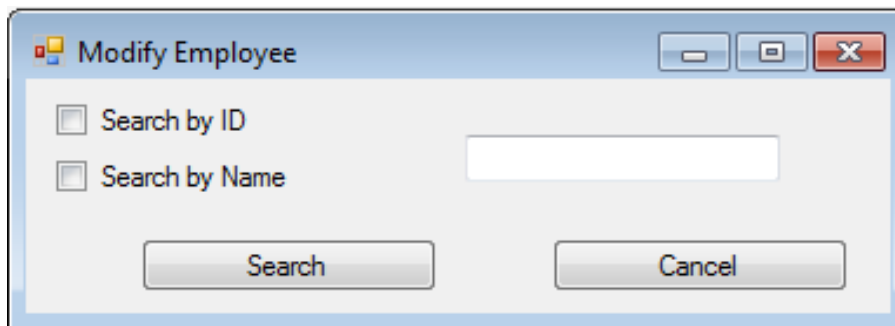
This is where a user searches for an employee to delete



A screenshot of a Windows-style dialog box titled "Delete Employee". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" and "Search by Name". To the right of these buttons is a text input field. Below the input field are two buttons: "Search" and "Cancel".

Modify Employee Search Page

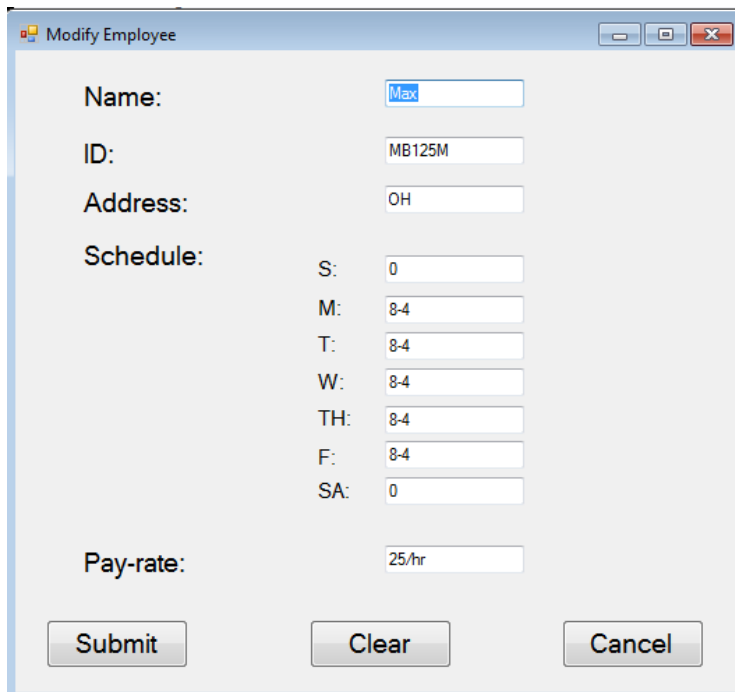
This is where a user searches for an employee to delete



A screenshot of a Windows-style dialog box titled "Modify Employee". The dialog has a standard title bar with minimize, maximize, and close buttons. Inside, there are two radio buttons: "Search by ID" and "Search by Name". To the right of these buttons is a text input field. Below the input field are two buttons: "Search" and "Cancel".

Modify Employee Page

This is where a user modifies an employee's info

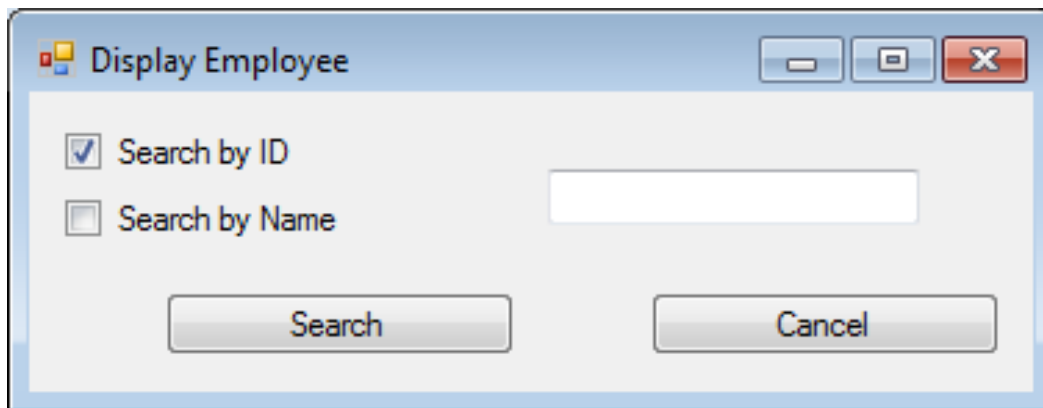


The 'Modify Employee' dialog box contains the following fields and controls:

- Name:** Text input field containing 'Max'.
- ID:** Text input field containing 'MB125M'.
- Address:** Text input field containing 'OH'.
- Schedule:** A group of seven text input fields for days of the week:
 - S: 0
 - M: 8-4
 - T: 8-4
 - W: 8-4
 - TH: 8-4
 - F: 8-4
 - SA: 0
- Pay-rate:** Text input field containing '25/hr'.
- Buttons:** 'Submit', 'Clear', and 'Cancel' buttons at the bottom.

Display an Employee Search Page

This is where the user searches for an employee to display

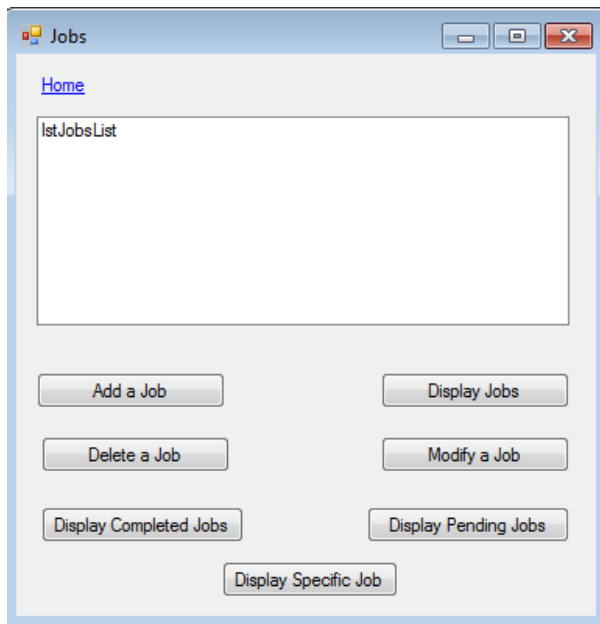


The 'Display Employee' dialog box contains the following fields and controls:

- Search by ID:** A checked checkbox.
- Search by Name:** An unchecked checkbox.
- Search Input:** A text input field for the search criteria.
- Buttons:** 'Search' and 'Cancel' buttons at the bottom.

Jobs Page

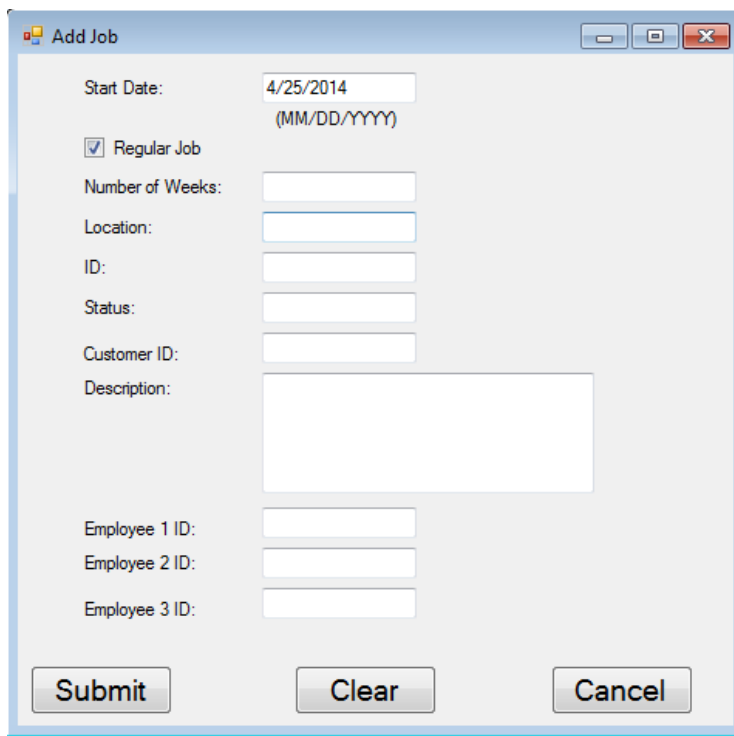
This is where a user sees all the jobs and performs various requirements to jobs



The screenshot shows a web application window titled "Jobs". It features a "Home" link at the top left. Below the link is a large, empty rectangular area labeled "lstJobsList". At the bottom of the window, there are six buttons arranged in three rows: "Add a Job", "Delete a Job", and "Display Completed Jobs" in the first row; "Display Jobs", "Modify a Job", and "Display Pending Jobs" in the second row; and "Display Specific Job" centered in the third row.

Add a Job Page

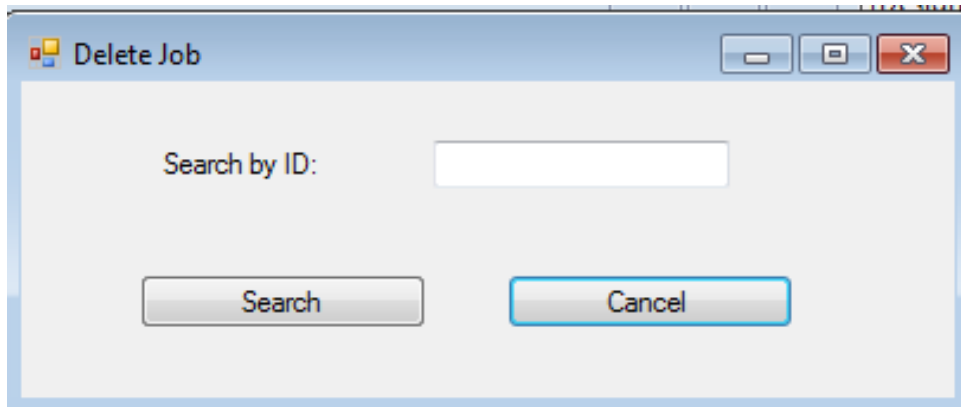
This is where a user adds a job



The screenshot shows a web application window titled "Add Job". It contains several input fields and a checkbox. The "Start Date:" field is pre-filled with "4/25/2014" and has a sub-label "(MM/DD/YYYY)". Below it is a checked checkbox labeled "Regular Job". The "Number of Weeks:" field is empty. The "Location:", "ID:", "Status:", and "Customer ID:" fields are all empty. The "Description:" field is a large, empty text area. At the bottom, there are three input fields for "Employee 1 ID:", "Employee 2 ID:", and "Employee 3 ID:". At the very bottom of the window are three buttons: "Submit", "Clear", and "Cancel".

Delete Job Search

This is where a user searches for a job to delete



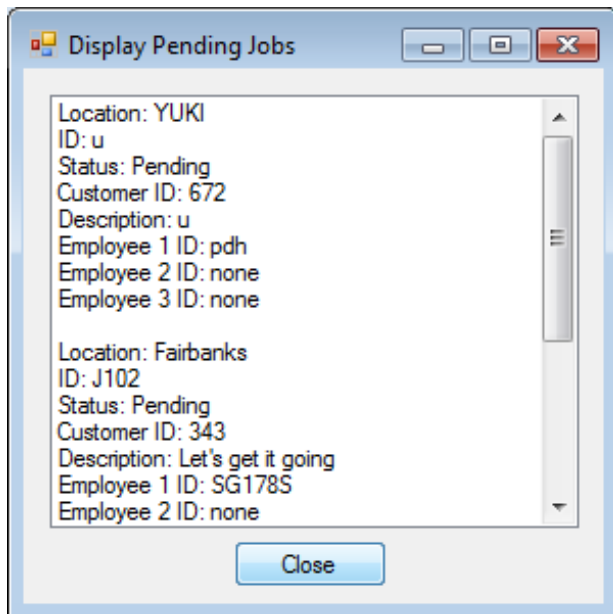
Display Completed Jobs Page

This is where a user looks at the completed jobs



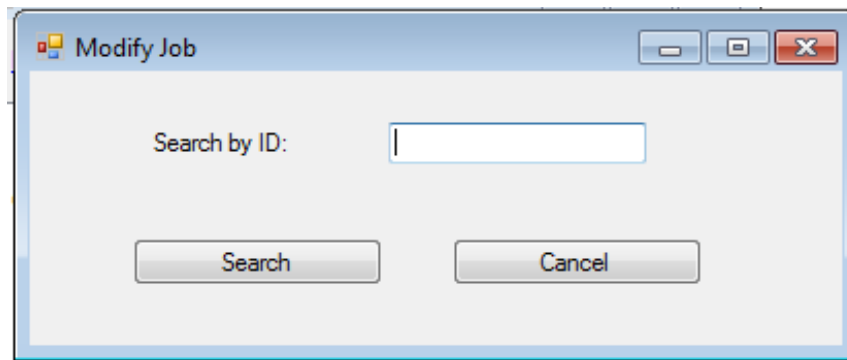
Pending Jobs Page

This is where a user looks at jobs not completed



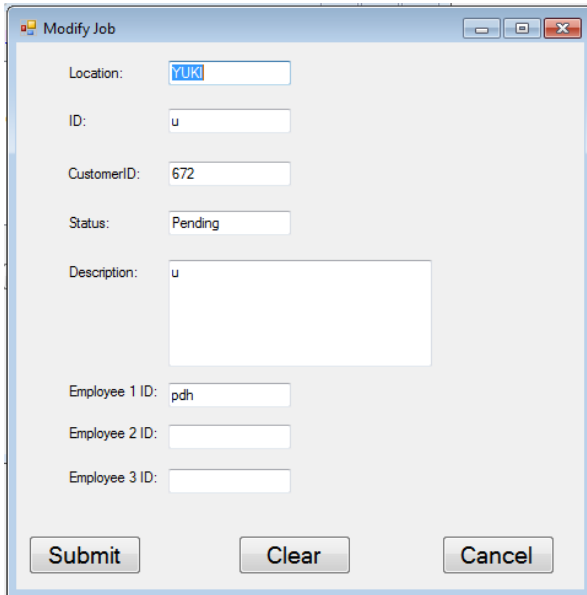
Modify a Job Search Page

This is where a user goes to search for a job to modify



Modify a Job Page

This is where a user modifies a job

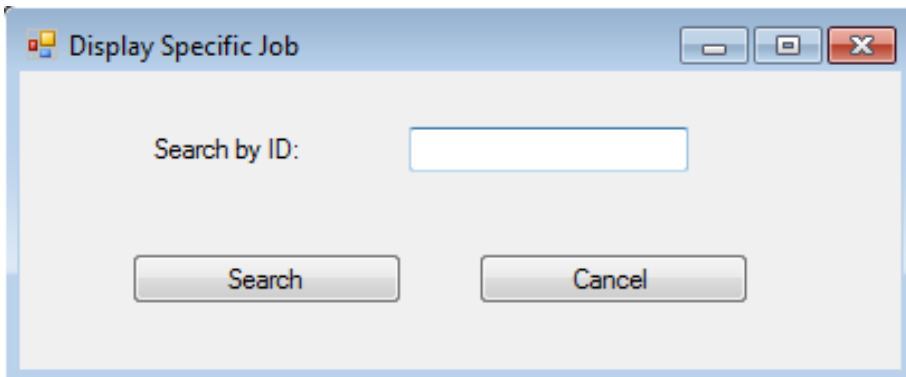


The 'Modify Job' dialog box contains the following fields and controls:

- Location:
- ID:
- CustomerID:
- Status:
- Description:
- Employee 1 ID:
- Employee 2 ID:
- Employee 3 ID:
- Buttons: Submit, Clear, Cancel

Display a Specific Job Search Page

This is where a user searches for a specific job to display

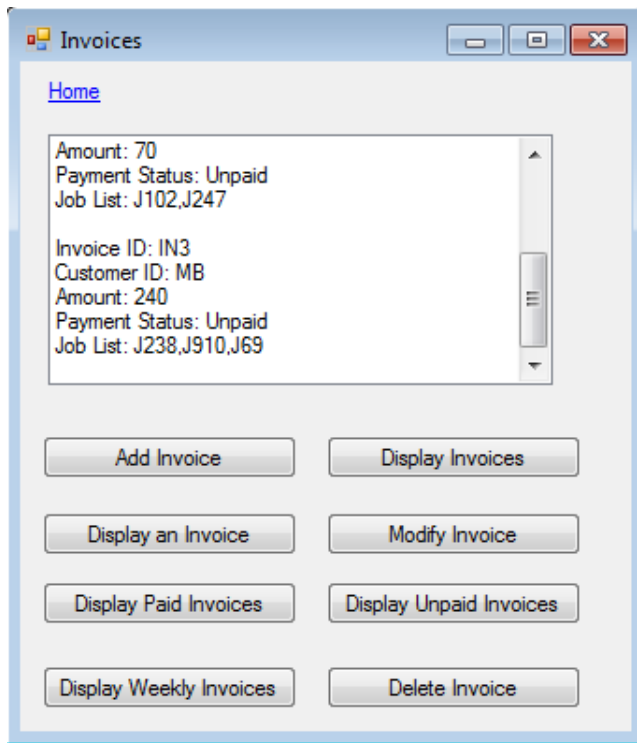


The 'Display Specific Job' dialog box contains the following fields and controls:

- Search by ID:
- Buttons: Search, Cancel

Invoice Page

This is where a user sees all the invoices and performs various requirements to invoices



The 'Invoices' window displays a list of invoices under a 'Home' tab. The list contains two entries:

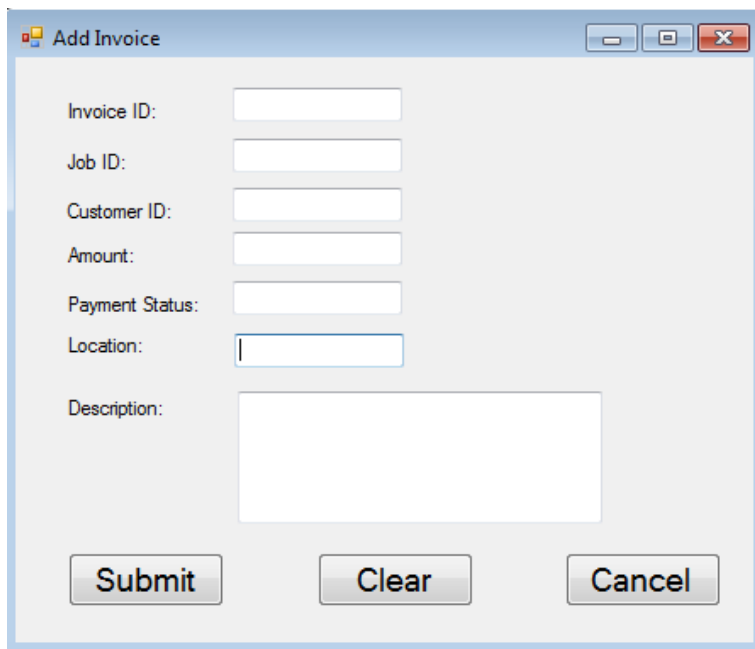
- Amount: 70
Payment Status: Unpaid
Job List: J102,J247
- Invoice ID: IN3
Customer ID: MB
Amount: 240
Payment Status: Unpaid
Job List: J238,J910,J69

Below the list, there are eight buttons arranged in four rows and two columns:

- Add Invoice
- Display Invoices
- Display an Invoice
- Modify Invoice
- Display Paid Invoices
- Display Unpaid Invoices
- Display Weekly Invoices
- Delete Invoice

Add an Invoice Page

This is where a user adds an invoice



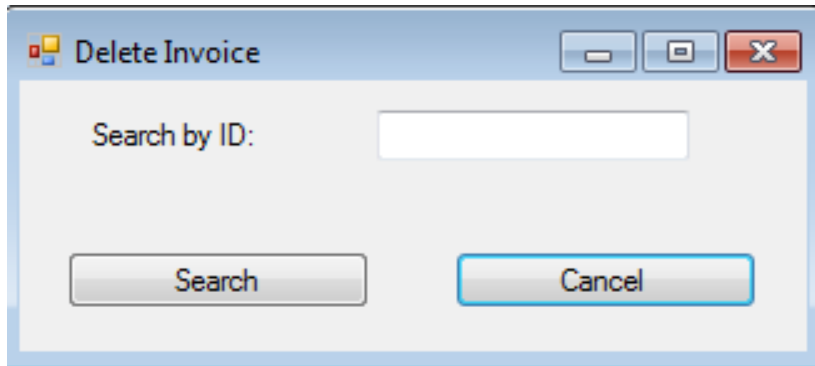
The 'Add Invoice' window contains the following input fields:

- Invoice ID:
- Job ID:
- Customer ID:
- Amount:
- Payment Status:
- Location:
- Description:

At the bottom, there are three buttons: Submit, Clear, and Cancel.

Delete an Invoice Search Page

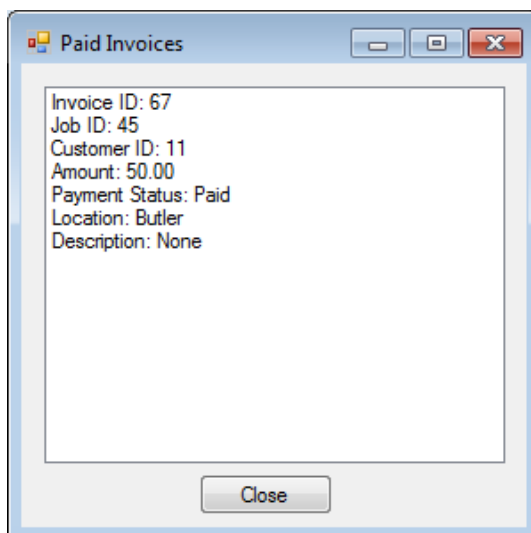
This is where a user searches for an invoice to delete



A dialog box titled "Delete Invoice" with a standard Windows-style title bar (minimize, maximize, close buttons). The main area contains a label "Search by ID:" followed by a text input field. Below the input field are two buttons: "Search" and "Cancel".

Display the Paid Invoices Page

This is where a user views the paid invoices



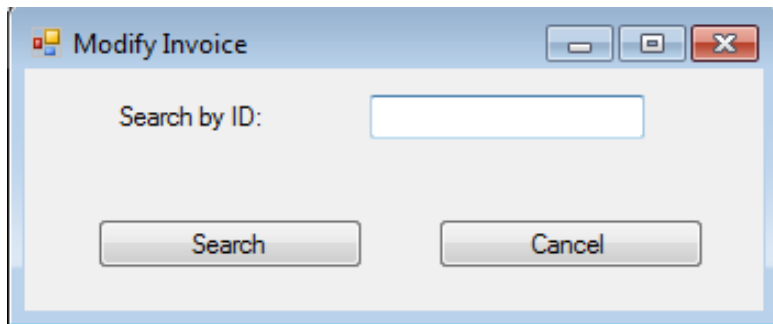
A dialog box titled "Paid Invoices" with a standard Windows-style title bar. The main area contains a list of invoice details:

- Invoice ID: 67
- Job ID: 45
- Customer ID: 11
- Amount: 50.00
- Payment Status: Paid
- Location: Butler
- Description: None

At the bottom of the dialog box is a "Close" button.

Modify an Invoice Search Page

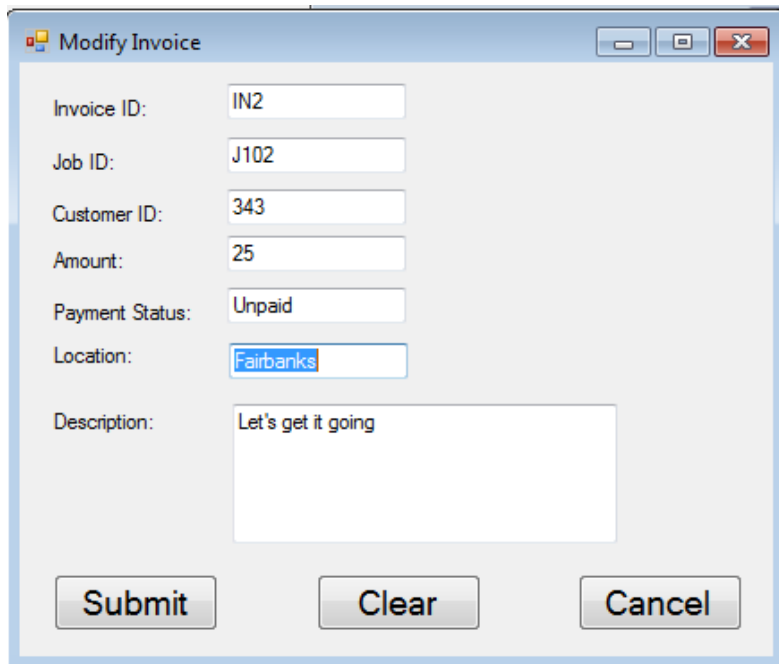
This is where a user searches for an invoice to modify



A screenshot of a web application dialog box titled "Modify Invoice". The dialog has a light blue header bar with standard window controls (minimize, maximize, close). Below the header, the text "Search by ID:" is followed by a text input field. At the bottom of the dialog, there are two buttons: "Search" and "Cancel".

Modify an Invoice Page

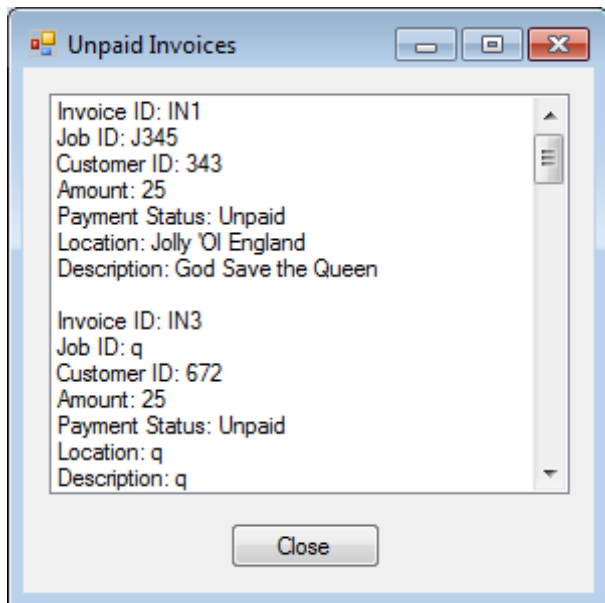
This is where a user modifies an invoice



A screenshot of a web application form titled "Modify Invoice". The form has a light blue header bar with standard window controls. Below the header, there are several input fields with labels to their left: "Invoice ID:" with the value "IN2", "Job ID:" with the value "J102", "Customer ID:" with the value "343", "Amount:" with the value "25", "Payment Status:" with the value "Unpaid", and "Location:" with the value "Fairbanks". Below these is a larger text area labeled "Description:" containing the text "Let's get it going". At the bottom of the form, there are three buttons: "Submit", "Clear", and "Cancel".

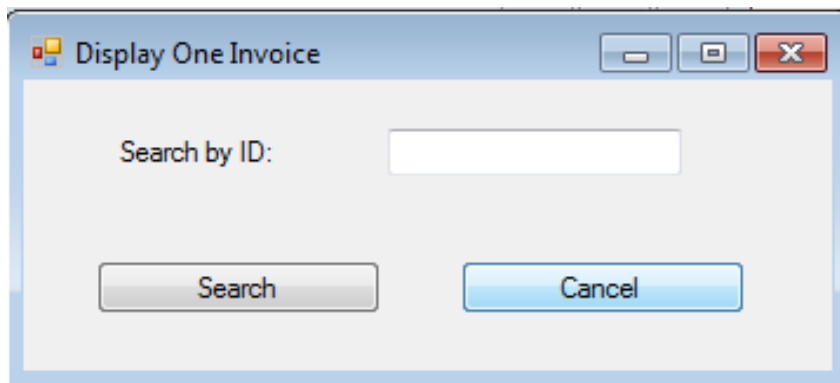
Unpaid Invoices Page

This is where a user views the unpaid invoices



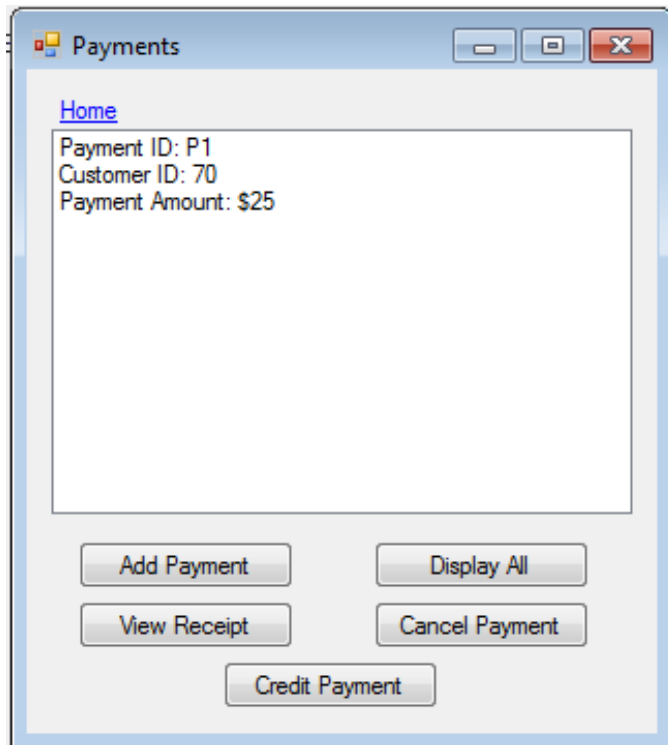
Display an Invoice Search Page

This is where a user searches for a specific invoice to display



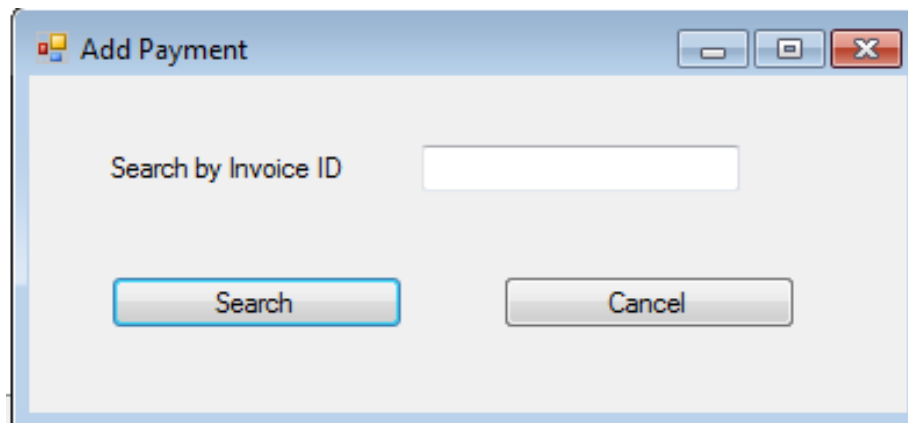
Payments Page

This is where a user sees all the payments and performs various requirements to payments



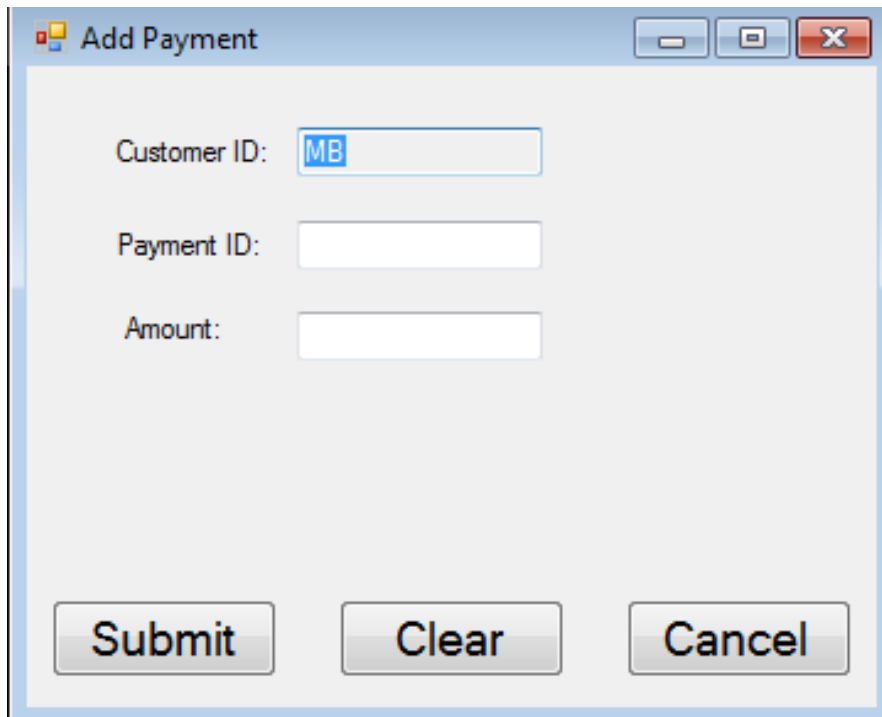
Add Payment Search Page

This is where a user searches for the invoice to add a payment to



Add Payment Page

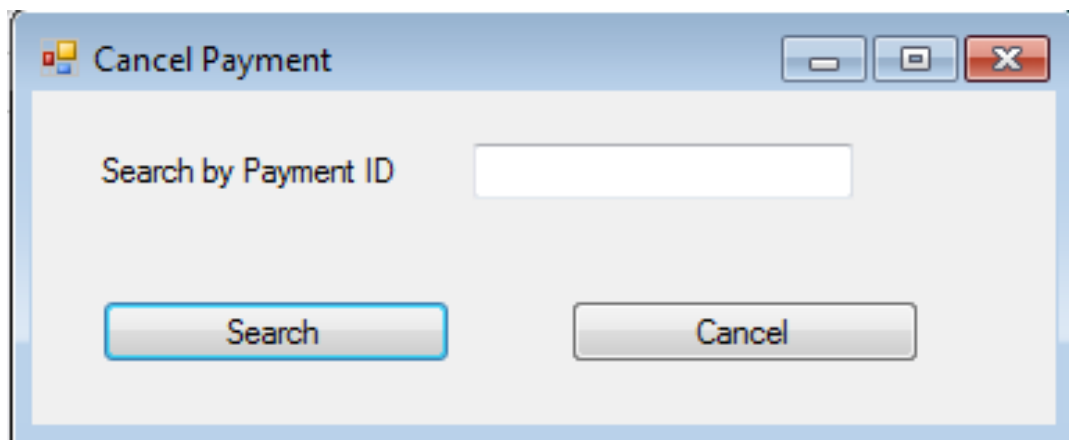
This is where a user adds payment to an invoice



A dialog box titled "Add Payment" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog contains three input fields: "Customer ID:" with the text "MB" entered, "Payment ID:" which is empty, and "Amount:" which is empty. At the bottom of the dialog are three buttons: "Submit", "Clear", and "Cancel".

Cancel Payment Search Page

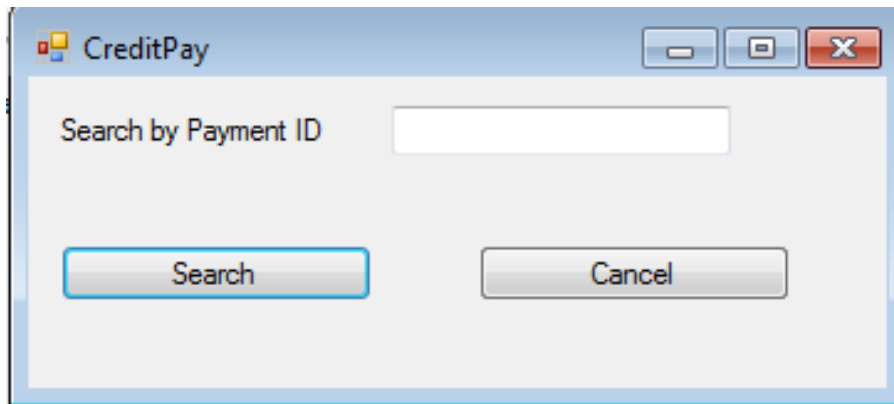
This is where a user searches to cancel a payment



A dialog box titled "Cancel Payment" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog contains a single input field labeled "Search by Payment ID". Below the input field are two buttons: "Search" and "Cancel".

Credit Payment Search Page

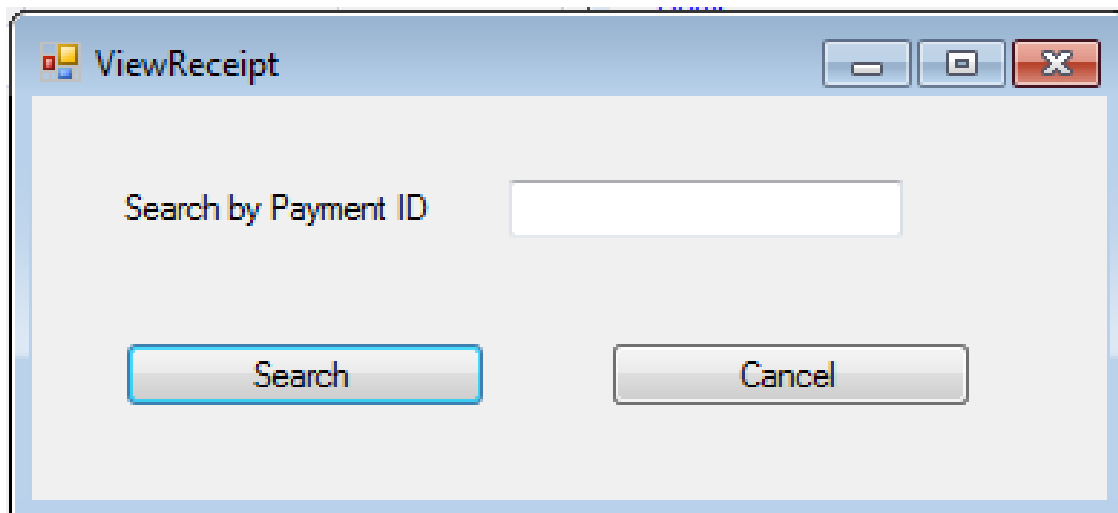
This is where a user searches to credit a payment



A screenshot of a Windows-style dialog box titled "CreditPay". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains the text "Search by Payment ID" followed by a white text input field. Below the input field are two buttons: "Search" and "Cancel".

View Receipt Search Page

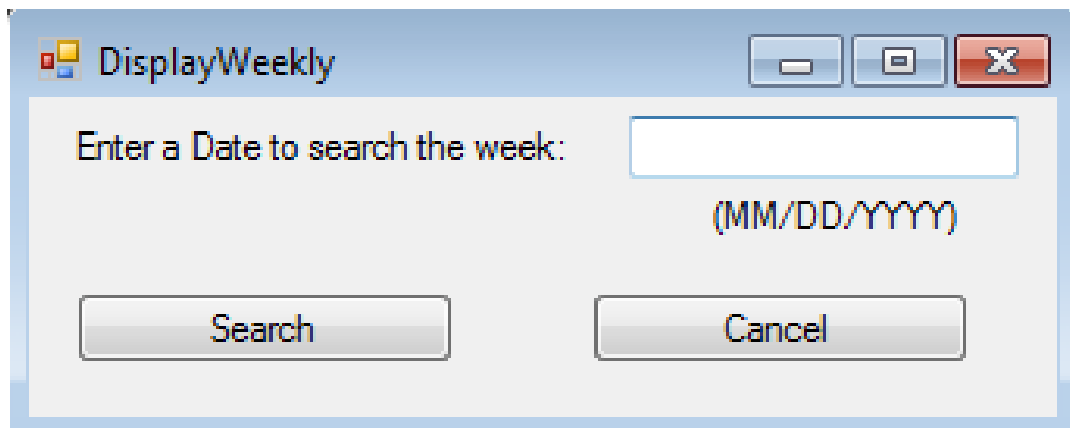
This is where a user searches for a receipt they want to see for a payment



A screenshot of a Windows-style dialog box titled "ViewReceipt". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains the text "Search by Payment ID" followed by a white text input field. Below the input field are two buttons: "Search" and "Cancel".

Display Weekly Invoices Search Page

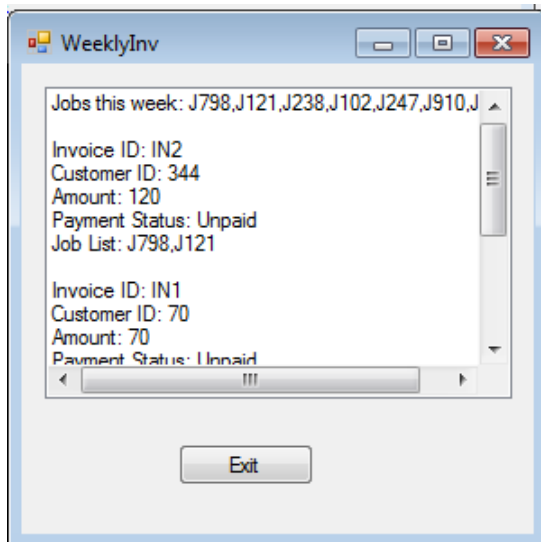
This is where a user searches for a week to view the invoices in that week



A screenshot of a Windows-style application window titled "DisplayWeekly". The window has a standard title bar with minimize, maximize, and close buttons. The main content area contains a label "Enter a Date to search the week:" followed by a text input field. Below the input field is a placeholder text "(MM/DD/YYYY)". At the bottom of the window are two buttons: "Search" and "Cancel".

Weekly Invoices Page

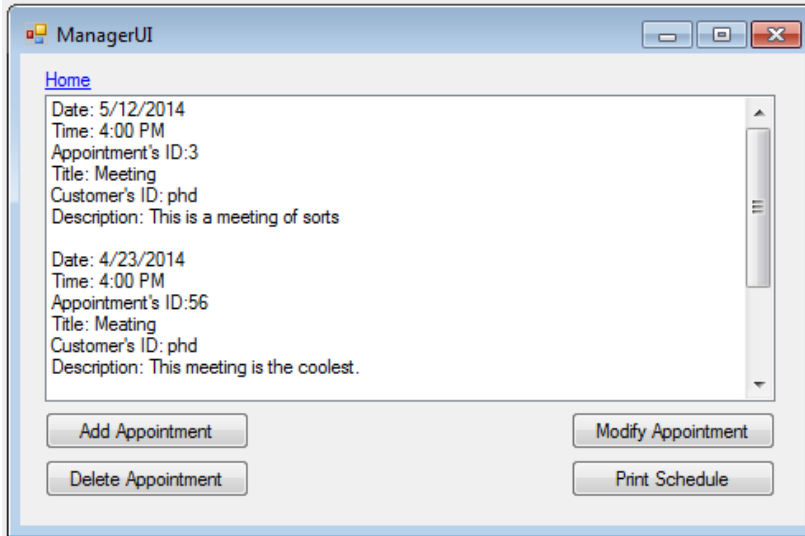
This is where the weekly invoices are displayed



A screenshot of a Windows-style application window titled "WeeklyInv". The window has a standard title bar with minimize, maximize, and close buttons. The main content area displays a list of jobs for the week: "Jobs this week: J798,J121,J238,J102,J247,J910,J". Below this, there are two sections of invoice details. The first section shows: "Invoice ID: IN2", "Customer ID: 344", "Amount: 120", "Payment Status: Unpaid", and "Job List: J798,J121". The second section shows: "Invoice ID: IN1", "Customer ID: 70", "Amount: 70", and "Payment Status: Unpaid". At the bottom of the window is an "Exit" button.

Manager Page

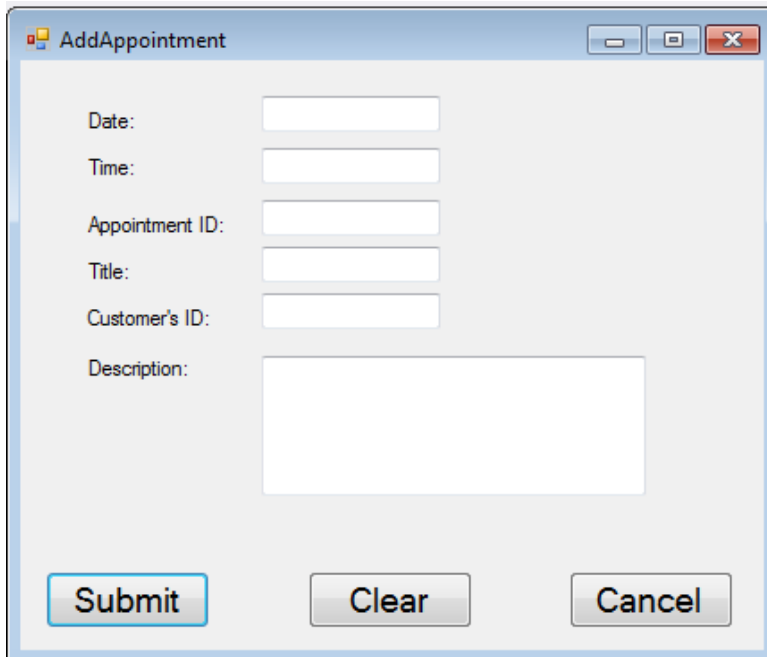
This is where a user sees all the appointments for the manager and performs various requirements for those appointments



The ManagerUI window displays a list of appointments. The first appointment is dated 5/12/2014 at 4:00 PM, with ID 3, titled 'Meeting', for customer 'phd', with the description 'This is a meeting of sorts'. The second appointment is dated 4/23/2014 at 4:00 PM, with ID 56, titled 'Meating' (sic), for customer 'phd', with the description 'This meeting is the coolest.' Below the list are four buttons: 'Add Appointment', 'Delete Appointment', 'Modify Appointment', and 'Print Schedule'.

Add Appointment

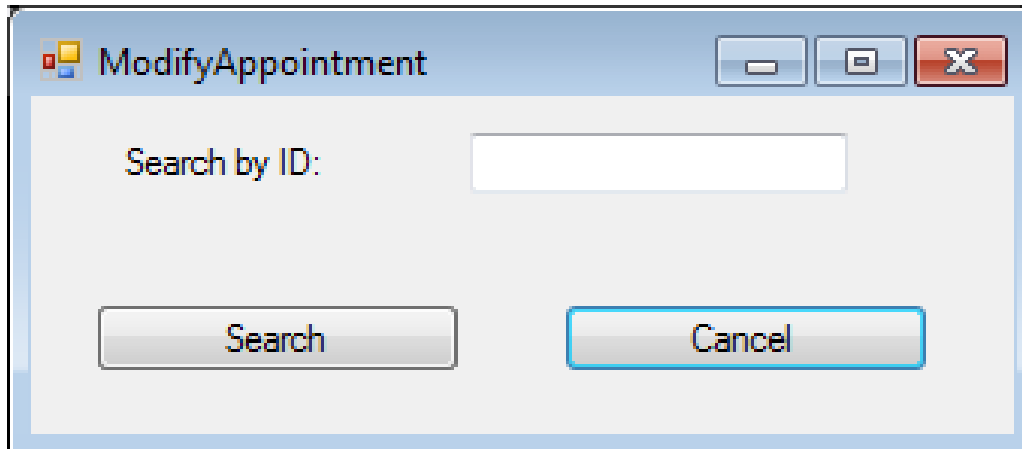
This is where a user adds an appointment



The AddAppointment window contains input fields for the following information: Date, Time, Appointment ID, Title, Customer's ID, and Description. At the bottom of the window are three buttons: 'Submit', 'Clear', and 'Cancel'.

Modify Appointment Search Page

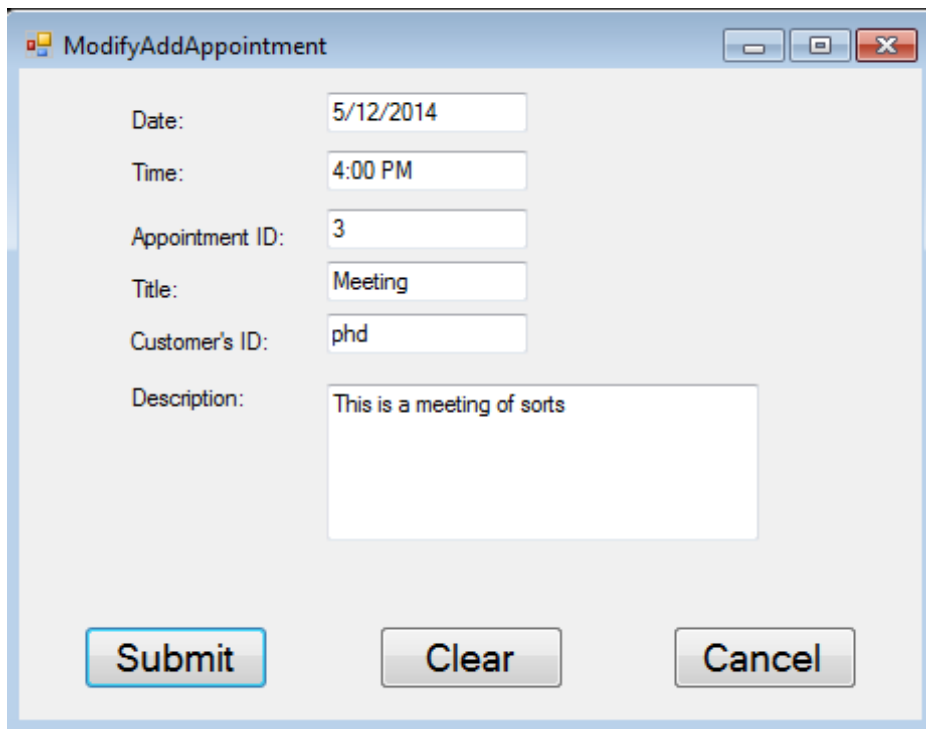
This is where a user searches for an appointment to modify



A screenshot of a Windows-style dialog box titled "ModifyAppointment". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains a label "Search by ID:" followed by a white text input field. Below the input field are two buttons: "Search" and "Cancel".

Modify Appointment Page

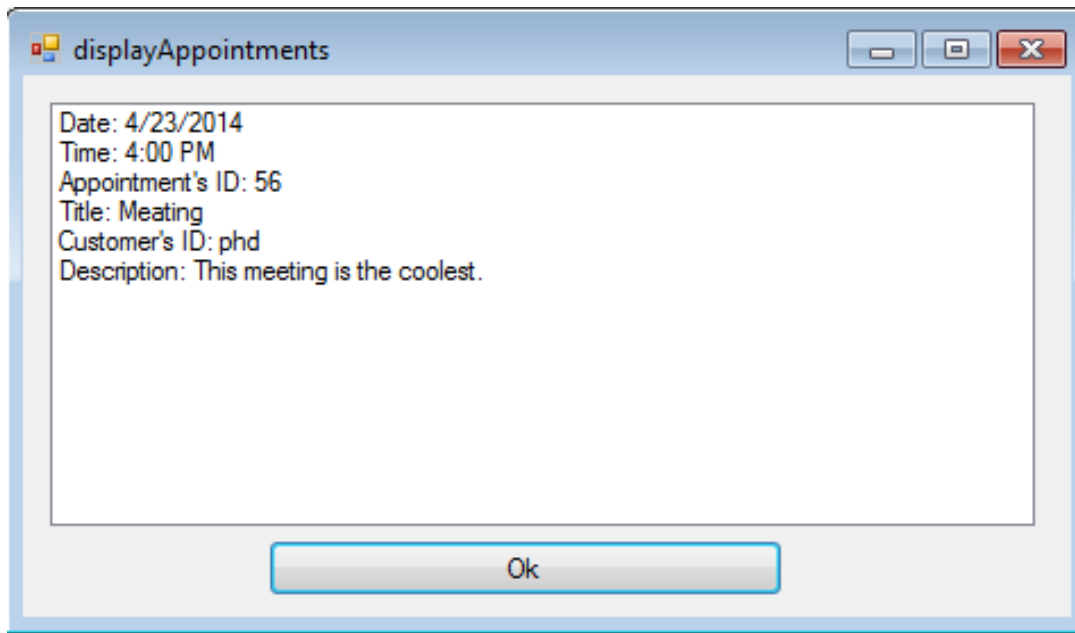
This is where a user modifies the appointment



A screenshot of a Windows-style dialog box titled "ModifyAddAppointment". The dialog has a light blue header bar with standard minimize, maximize, and close buttons. The main area is light gray and contains several form fields: "Date:" with a date picker showing "5/12/2014", "Time:" with a time picker showing "4:00 PM", "Appointment ID:" with a text input field containing "3", "Title:" with a text input field containing "Meeting", "Customer's ID:" with a text input field containing "phd", and "Description:" with a larger text area containing "This is a meeting of sorts". At the bottom of the dialog are three buttons: "Submit", "Clear", and "Cancel".

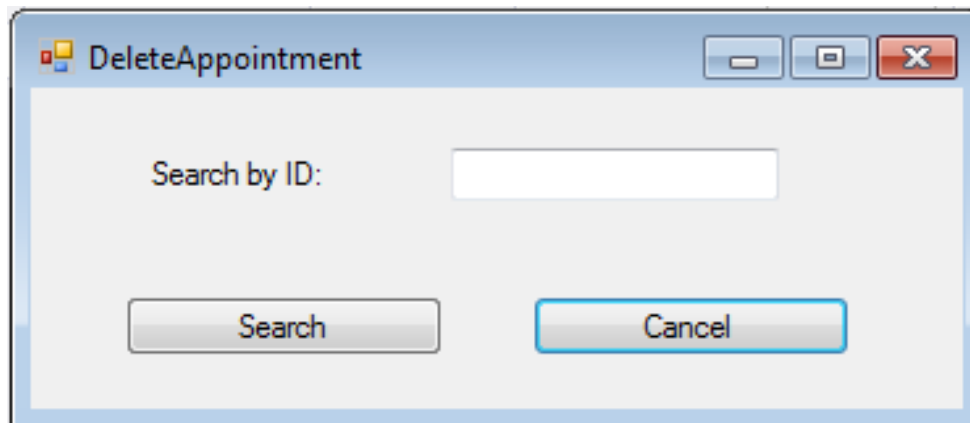
Display Appointments

This is where the appointments for that week are displayed



Delete Appointment Page

This is where a user searches for an appointment to delete



Flexibility of Our Design

When our team first began to design the basic layout of how we wanted our classes and code organized, we had a limited knowledge about software architecture. We threw together some entity classes and had a general idea of what we wanted, but our design was far from being polished. Transitioning from Iteration 1 to Iteration 2 was a huge wake up call, as more and more functionality was being required. This forced us to really improve upon our class organization and structure. By the end of Iteration 2, our team had put together a class diagram we were proud of. The layering of entities, boundaries, and controllers modeled a professional piece of software, and this our design became more flexible. We hit the ground running in Iteration 3, as the addition of new classes became intuitive given our solid foundation. Overall, the flexibility of our design allowed us to work efficiently in the final stages of this project, which was a huge accomplishment for us.

Tested Code

For our completed application, please refer to our Wiki. Over the course of this iteration, we hand tested our code, as we did not learn about unit testing yet. We felt that hand testing was sufficient, given the scope of this project but unit testing would be necessary if this application was extended to handle a large amount of data.

Chapter 4: Conclusions

The experiences of the software development approaches have really altered our view on software development. When we began this project, we were unaware of the amount of preparation we would need to code. This was the first time that we were exposed to an iterative approach; it was slower than it would normally be due to the lack of exposure. The hardest part was our understanding of all these new subjects along the way as we did not fully understand them until later in the project.

If we knew how to make a proper class diagram, proper interaction diagrams and other proper documentation before this class and project, we could have avoided some of the headaches we encountered during the first iteration especially. In the professional world of software engineering, we would have developed our UML diagrams to focus on much more detail. For the code, it was a similar problem. As our knowledge of Visual Basic grew, our GUI and structure of code improved as a whole.

With this lack of knowledge and experience, we required more time for development especially as we started new iterations. We were not only adding on to this software, but also restructuring our code to be better and more object-oriented. We thought that overall this is an efficient process to build real, professional software, but there is a serious need for experienced programmers and software engineers to direct workflow. If done correctly the agile and iterative approach is very powerful, but needs to be planned and directed correctly.

Using the object-oriented approach, the main advantage is that we can develop our entities and make them very efficient for writing to files in the back end. It is also very easy to understand the flow of the project by creating controllers that are objects. These controllers are very easy to work with in the code. One of the disadvantages in the object-oriented approach is that there is a possibility of serious restructuring if new requirements are introduced. Another disadvantage is with the new entities there could easily be a lot of overhead work by adding new controllers and/or new boundaries. Overall, the object-oriented approach is a great way to develop professional software.

Chapter 5: Team Organization and Roles

The team was a single level team with not much leadership as we held each other accountable versus a hierarchical group. However there were still roles and contributions for each person.

The team roles were as follows:

Philip Dwyer

He was the official team lead by completing all the weekly status reports. He was the documentation lead as well by organizing most things that were not code related. The sequence diagrams were handle by him, as well as other documentation. He updated the TOE tables to handle the 2nd iteration. In general, Phil handled the Wiki, so the errors in the Wiki were his responsibility.

Adam Sanders

He was the coding lead, which involved handling the majority of the code. Towards the end of this project, Adam became more of the team lead as the last two weeks of the project involved mostly code. With his duties, he molded into a vice team lead by organizing and informing our group of upcoming deadlines and due dates. Also, in the early stages of the project, Adam contributed to the diagrams.

Max Brodbeck

He was the Dabbler or the “everything else” lead, which consisted of handling some of the code, completing some of the diagrams, and performing some of the documentation. He wrote all of the Weekly Schedule requirements code, redesigned some of the GUI, and worked on the Invoices requirements for code. Also he updated the CRC cards and originally created the TOE tables. He originally created the class diagrams as well.

Sean Gibbens

He was the diagram lead, which was the handling of the final class diagram and any other diagram that spawned from the class diagram like the package diagram. Sean also handled the use case diagram organizing it completely in the early stages of the project. In terms of coding, Sean contributed to the Jobs section in our software.

Appendices

Peer Evaluation Forms

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. **The GUI design was well modeled and presented.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. **The overall presentation was well prepared and delivered by the team effectively.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. **The team appears to be well organized and functioning.**
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. **I have the following additional constructive suggestions and/or comments for the team:**

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

Great Job!

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree) *need work*
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree) *need work*
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree) *need work*
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

~~ITERATION~~ I

~~TEAM NAME~~

SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

Lots of "umm"s

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. The interaction diagrams seemed to be complete and represented the message passing among classes.

(Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. The GUI design was well modeled and presented.

(Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

8. The overall presentation was well prepared and delivered by the team effectively.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. The team appears to be well organized and functioning.

(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. I have the following additional constructive suggestions and/or comments for the team:

TEXT AS BUTTONS SHOULD BE USED SPARINGLY

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME:

SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPDA

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. The interaction diagrams seemed to be complete and represented the message passing among classes.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

6. The GUI design was well modeled and presented.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. The overall presentation was well prepared and delivered by the team effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. The team appears to be well organized and functioning.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. I have the following additional constructive suggestions and/or comments for the team:

Interaction Diagram - yours kind of confused me. Like each connection arrow should be a function call and yours didn't seem that way.

GUI - I think you could simplify and not have to open so many windows to get where you want to.

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Spam

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

Nice log-in !

Fresco, Good Stuff

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demonstration was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and represented the message passing among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The GUI design was well modeled and presented.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The team appears to be well organized and functioning.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

~~Team~~ TEAM SPAM

April 17

- 1 Demo completeness: 5
- 2 Use case diagram / Scenarios improvement: 4
- 3 CRC cards: 3 didn't really go over them turned me on
- 4 Interaction diagrams: 4 ↙
- 5 UML class completeness: 5 beautiful class diagram - so clean and organized
- 6 Layers presented well: 5
- 7 GUI improvement: 4 great improvement but still room to improve more
- 8 UML design mapping to code: 4.9
- 9 Overall presentation: 5

Comments

- It would be nice to be able to click on a customer in the list to display them
- I like that invoices are auto created for jobs
- It'd be more user friendly to not have to open a new form to view by payment status / completion status
- (payed == paid) = true
- Nice doge
- Should NOT have to change ID to modify an entity
- Huge GUI improvement. Wow. Such GUI. obj Nice.
- Team SPAM went ham on iteration 2

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
2. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
3. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 (4) (5) (Strongly Agree)
4. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
5. The UML class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
6. All the Entity, Boundary and Control classes were presented and explained well.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The UML class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. All the Entity, Boundary and Control classes were presented and explained well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2

TEAM NAME:

SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The UML class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. All the Entity, Boundary and Control classes were presented and explained well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. I have the following additional constructive suggestions and/or comments for the team:

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
2. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
3. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
4. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)
5. The UML class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
6. All the Entity, Boundary and Control classes were presented and explained well.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 (5) Strongly Agree
10. I have the following additional constructive suggestions and/or comments for the team:

— GUI can be polished
— Sequence diagrams can be more detailed (loops, decisions)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #:

TEAM NAME: TEAM SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. All defects detected and reported to the developers were addressed effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree) *could have shown interaction or branching*

6. The class diagram included all attributes, operations and relationships among classes.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. The GUI design was improved and presented well.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. The overall presentation was well prepared and delivered by the team effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. The team appeared to be better organized and improved since the beginning of the semester.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

Nice job!

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3

TEAM NAME:

Spam

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. All defects detected and reported to the developers were addressed effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. The team appeared to be better organized and improved since the beginning of the semester.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

Good Job Bro's
Get some version control

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #:

TEAM NAME:

SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. All defects detected and reported to the developers were addressed effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. The team appeared to be better organized and improved since the beginning of the semester.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

such presentation
very wow

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3

TEAM NAME: SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. All defects detected and reported to the developers were addressed effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. The team appeared to be better organized and improved since the beginning of the semester.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

Good Job!

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #:

3 SPAM

TEAM NAME:

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. All defects detected and reported to the developers were addressed effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

6. The class diagram included all attributes, operations and relationships among classes.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. The GUI design was improved and presented well.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. The overall presentation was well prepared and delivered by the team effectively.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. The team appeared to be better organized and improved since the beginning of the semester.

(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #:

TEAM NAME:

SPAM

Please rate the team's work and presentation by circling the right number.

1. The demo was complete and included all the new expected functionality for this iteration.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
2. All defects detected and reported to the developers were addressed effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
3. The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
4. The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
5. The interaction diagrams seemed to be complete and presented the message passing among classes effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
6. The class diagram included all attributes, operations and relationships among classes.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
7. The GUI design was improved and presented well.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
8. The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
9. The overall presentation was well prepared and delivered by the team effectively.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)
10. The team appeared to be better organized and improved since the beginning of the semester.
(Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

Final CRC Cards

These are the final CRC Cards for all iterations with all six controllers and entities.

Customer <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide Customer Info (Name, Phone, Address, Repeat, ID)	

Job <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Store job info (When, Where, ID, Employees Assigned)	Employee

Invoice <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Store invoice info like customer ID, Job ID, and whether it is paid or not	Customer, Job

Employee <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide information about employee	

Appointment <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide information on the appointment like date and time	Customer

Payment <Entity>	
<i>Responsibility</i>	<i>Collaborator</i>
Provide information on the payment like customer and amount	Customer

CustomerController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
Add Customer	Customer
Load Customer	Customer
Check ID if exists	Customer
Modify Customer	Customer
Delete Customer	Customer
Display Customer	Customer

EmployeeController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
Add Employee	Customer, Employee
Load Employee	Employee
Check ID if exists	Customer, Employee
Modify Employee	Customer, Employee
Delete Employee	Employee
Display Employee	Employee

JobController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Job	Customer, Employee, Job, Invoice
-Load Jobs	Job
-Check to see if ID exists for customer and employee	Customer, Employee
-Modify Job	Customer, Employee, Job, Invoice
-Delete Job	Job
-Display All Jobs, a Job, or jobs in a week	Job
-check to see what invoices are worked on for the week	Job, Invoice

InvoiceController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Invoice	Customer, Job, Invoice
-Load Invoice	Job
-Edit Invoices when modifying or adding Jobs	Job, Invoice
-Check to see if IDs exist and get IDs	Customer, Job, Invoice
-Apply Payments to Invoices	Customer, Invoice, Job
-Delete Invoice	Invoice
-Display Invoices As All or One	Invoice
-Create Invoice ID	Invoice

PaymentController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Payment	Customer, Payment
-Load Payment	Payment
-Cancel Payment	Payment
-Credit Payment	Payment
-Display A Payment	Payment

ManagerController <Controller>	
<i>Responsibility</i>	<i>Collaborator</i>
-Add Appointment	Customer, Appointment
-Load Schedule	Appointment
-Display Schedule	Appointment
-Delete Appointment	Appointment
-Modify Appointment	Customer, Appointment
-print Weekly Schedule	Appointment

Final Use-case Diagram

This is the final use-case diagram that shows all three iterations' completed work. For a closer look, please refer to our wiki on Moodle.

