# General Notes

* Each Case Study will have a facilitator assigned to it as a Product Owner.
  + The Product Owner makes all final decisions on questions dealing with the case study.
  + The Product Owner can decide to change or modify user stories, as long as the scope is not expanded.
  + The Product Owner can invalidate / remove user stories. Such stories will not be considered for the purposes of final scoring and evaluation.
  + The Product Owner cannot add user stories after development has started.
* Unless otherwise specified, data should be persisted to a database.
* Unless otherwise stated, each story with a UI should have its own screen.
* A general acceptance criterion is that all functional and validation requirements of the story are met.

# Implementation Hints

* Use BigDecimal for currencies since using float or double may result in bugs when dividing or multiplying
* Use StringUtils from Apache Commons library to check for null/empty/blank strings.

# Case Study 1: Employment System

## General Description

The Employment System application is used by an HR Manager to track employee compensation. Different types of compensation are tracked such as monthly salary, bonuses, commission, etc.

## User Stories

### User Story 1.1: View Home Page

#### Description

As an HR Manager, I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 1.2: Add Employee

#### Description

As an HR Manager, I want to add an employee so that I can manage employee information.

#### Functional Requirements

Employee profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Position | String (free form) | Yes |
|  |  |  |

In addition, the system should generate a UID for each successfully added employee.

**UI**

1. Employee entry form should have its own screen.
2. User should receive feedback that an employee was successfully added.
3. User should receive feedback if an employee was not added for the following reasons:
   1. Employee already exists – User should be alerted
   2. Missing required fields – User should be alerted as to which fields are missing
   3. Invalid field values – User should be alerted as to why a field value is wrong
4. If employee was not added for some other reason, a generic error message should be displayed

**Validation**

1. An employee cannot be entered into the system twice. Employee identity is based on first name, middle name, last name, and birth date. Employees with all three identical fields are considered to be the same employee.
2. Birth date should not be later than current date.

#### Out-of-scope

1. Legal age validation

#### Dependencies

None

#### Acceptance Criteria

1. User can add employee to the system as long as it passes validation.
2. All functional requirements are met.

### User Story 1.3: Search Employees

#### Description

As an HR Manager I want to search for employees in the system so that I can select employee to manage.

#### Functional Requirements

Search by first name, last name, position or any combination of the three. Search result should match all supplied search criteria (AND).

UI

1. One field for each search criteria.
2. One Search button.
3. One Clear button to clear search criteria
4. Executing a search should not clear search criteria
5. Search results should display employee UID plus all fields specified in Story 1.2
6. If no matching results are found, “0 results found” should be displayed

#### Out-of-scope

N/A

#### Dependencies

Story is dependent on Story 1.2 Add Employee unless employee entities are already in the database.

#### Acceptance Criteria

1. User should be able to search for employees
2. All other functional requirements are fulfilled

### User Story 1.4: View/Edit Employee Profile

#### Description

As an HR Manager, I want to view and edit employee profiles so that I can change employee details when needed.

#### Functional Requirements

All employee fields except UID should be editable. Validation rules for employee fields from Story 1.2 must still be followed

UI

1. Employee to be viewed/edited must be selected from search results (see Story 1.3)
2. View/Edit Employee form must be in its own screen.
3. View/Edit form fields must be pre-populated with employee information.
4. All employee fields should be visible, including UID.
5. UID should only be displayed. It should not be editable.
6. User should receive feedback if employee was successfully edited.
7. User should receive feedback if an employee was not updated for the following reasons:
   1. Identical employee found – User should be alerted if editing an employee would make it identical to an existing employee
   2. Missing required fields – User should be alerted as to which fields are missing
   3. Invalid field values – User should be alerted as to why a field value is wrong
8. If employee was not edited for some other reason, a generic error message should be displayed
9. If employee was not edited, form fields should not be reset.

#### Out-of-scope

1. Legal age validation

#### Dependencies

Story 1.3

#### Acceptance Criteria

1. User can select an employee to view/edit from search results.
2. User can view/edit an employee.
3. All functional requirements are fulfilled.

### User Story 1.5: Add Employee Compensation Details

#### Description

As an HR Manager I want to add monthly compensation details for an employee so that I can track an employee’s work and salary in the system.

It is possible for an employee to receive multiple types of compensation in a month. For example, salespeople receive a monthly salary and can receive commission on sales and bonuses. These should be recorded in the system. Compensation details include Compensation type (salary, bonus, commission, etc.), amount, and description.

#### Functional Requirements

Assumptions

1. All amounts are in a single home currency where the application will be used.
2. It will be up to HR Manager to define monthly compensation (before or after taxes, SSS, etc.)

Fields

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Type | String | *See below* |
| Amount | Numeric | *See below* |
| Description | String | *See below* |
| Date | Date | Only year and month is important. Assume end-of-month pay period. |

Types of Compensation

1. Salary – This is basic monthly salary. Only one salary entry per employee per month can be added. It is possible for the value to be zero or negative due to deductions. Description is optional.
2. Bonus – Any bonus received beyond contractual salary. An employee can have multiple bonuses in a month. Amount must be greater than zero. Description is required.
3. Commission – Any compensation based on a percentage from closing sales. An employee can have multiple commission from multiple sales. Amount should be actual amount received, not percentage. Amount should be greater than zero. Description is required. Entry can be based on individual closed sales, or aggregated sales.
4. Allowance – This covers per-diem, over-time food or transportation allowance, etc. An employee can have more than one type of allowance. Amount should be greater than zero. Amount is assumed to have been converted to home currency. Description is required.
5. Adjustment – This covers anything not covered above. More than one entry is allowed. Amount can be any value except zero. Description is required.

Validation

1. See section above for validation rules for the different compensation types.

#### Out-of-scope

1. Compensation calculation based on employee position
2. Currency conversion for per-diem
3. Calculations of deductions (taxes, SSS, health insurance, etc.)
4. Editing of compensation details is another story and is out of scope for this story

#### Dependencies

Story 1.3

#### Acceptance Criteria

1. User can add compensation details as per functional requirements

### User Story 1.6: View Compensation History

#### Description

As an HR Manager, I want to view the monthly compensation history of an employee so that I can generate a report on how much an employee earns every month.

Only total compensation per month will be displayed for chosen employee.

#### Functional Requirements

UI Flow

1. User searches for employee
2. User selects employee
3. User enters start month/year and end month/year
4. New screen shows total compensation per month for each year
5. User can choose to return to Home Page or Step 4

UI

1. Total per month should be shown. No breakdown as this is done in another story.
2. If specified date range covers more than one year, show monthly compensation for each month in each year within specified date range. For example, do not add totals for January from all years together.
3. If no compensation details for a month can be found, that month is not displayed. However, in the case where compensation details total zero, that month should still be displayed.

Validation

1. When entering date range, user cannot enter an end date that occurs before start date.

#### Out-of-scope

Breakdown of monthly compensation is done in Story 1.7.

#### Dependencies

Story 1.3

#### Acceptance Criteria

1. Compensation report for an employee is displayed

### User Story 1.7: View Compensation Breakdown for a Month

#### Description

As an HR Manager I want to view monthly breakdown employment details for an employee so that I can see how much an employee is making from different types of income.

#### Functional Requirements

UI Flow

1. User brings up compensation history as per Story 1.6.
2. User selects displayed month from report.
3. New screen displays compensation details for that month.
4. User can go to home page or select new month to display.

UI

1. Only one month can be selected at a time for breakdown report
2. Total for the month should still be displayed in breakdown screen
3. All fields for compensation details should be displayed. Description can be truncated if too long.

#### Out-of-scope

None

#### Dependencies

Story 1.6

#### Acceptance Criteria

1. User can bring up compensation breakdown of an employee for a specific month.

### User Story 1.8: Edit Employee Compensation Details

#### Description

As an HR Manager I want to edit monthly employment details for an employee so that I can make changes to an employee’s work and salary record.

#### Functional Requirements

UI Flow

1. User brings up compensation breakdown as per Story 1.7
2. User selects compensation to edit
3. User edits compensation details
4. Compensation breakdown screen is displayed with updated details and totals.

UI

1. User can only edit amount and description.
2. Validation rules for adding compensation in Story 1.5 are still enforced

#### Out-of-scope

1. Deletion of compensation
2. Editing of compensation type and date.

#### Dependencies

Story 1.7

#### Acceptance Criteria

1. User is able to edit compensation details.

# Case Study 2: Trucking System

## General Description

The Trucking System is used by a Dispatcher to manage trucks and to dispatch trucks on delivery jobs. Status of trucks and jobs are kept in the system.

## User Stories

### User Story 2.1: View Home Page

#### Description

As a Dispatcher, I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 2.2: Add Truck

#### Description

As a Dispatcher, I want to add truck details so that I can manage my trucks.

UI Flow:

1. User clicks on “Add Truck” link on front page.
2. User is shown empty form.
3. User enters form data and clicks on submit button to submit.

Truck fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Truck ID | Any | System Generated upon submission |
| Model | String | Yes |
| License No. | String | Yes |
| Weight | Numeric > 0 | Yes |
| Capacity | Numeric >= 0 | Yes |
| Date Acquired | Date | Yes |
|  |  |  |

Validation Rules:

1. License No. should be unique. System must not accept license plate number if it us already being used by another truck. This can be implemented in UI, back-end, or both.
2. Date Acquired must be before or same as current date.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to add a truck in the system
2. Validation should be enforced

### User Story 2.3: List Trucks

#### Description

As a Dispatcher, I want to view a list of trucks in the system so that I can have an overall view of the trucks I have.

UI Flow:

1. User clicks on “List Trucks” link.
2. User is shown list of all trucks in a table showing all fields defined in story 2.2, including Truck ID

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should see list of trucks in the system
2. Required fields should be shown

### User Story 2.4: View/Edit Truck Details

#### Description

As a Dispatcher I want to view/edit truck details when I select a truck displayed in the list of trucks so that I can manage a specific truck.

UI Flow:

1. User selects a truck to edit from List Trucks screen.
2. User sees form with fields prepopulated with data from selected truck
3. User can edit form and click submit to update truck details
4. User receives feedback that truck was updated
5. User can cancel form if user only wishes to view truck details.
6. User is brought back to home page after editing truck or canceling form.

Functional Requirements:

1. Validation – Validation rules in Story 2.2 should be followed
2. Truck Id cannot be changed

#### Out-of-scope

Deletion of trucks

#### Dependencies

Story 2.3

#### Acceptance Criteria

1. User should be able to select a truck to view/edit from list trucks screen
2. User should be able to view truck details in form
3. User should be able to edit truck details and submit
4. User should be able to go back to home page after editing truck, or without editing truck
5. All validation rules should be followed

### User Story 2.5: Create Trip Schedule

#### Description

As a Dispatcher, I want to create trip schedules for my trucks so that I can schedule my trucks for jobs.

Trip Schedule fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Trip ID | Any | System Generated after submission |
| Truck ID | Any | From Truck Table |
| Driver Name | String | Yes |
| Load Weight | Numeric | Yes |
| Date/Time Started | Date/Time | Yes |
| Date/Time End | Date/Time | No |
| Status | Preset: 0 – Ready 1 – Loading  2 – In Transit  3 – Arrived/Unloading  4 – Unloaded  5 – Return Trip  6 – Returned  7 – Cancelled | Yes |

UI Flow:

1. User selects truck from truck list view.
2. User clicks on “Schedule Trip”
3. Trip form is displayed with Truck ID prepopulated
4. User submits or cancels form
5. User is brought back to home page after successful submission or cancellation of trip form.

Validation:

1. Date/Time End if supplied cannot be before Date/Time Started
2. Load weight cannot be greater that truck’s capacity

#### Out-of-scope

Checking for schedule conflicts, for example drivers or trucks with overlapping schedules.

#### Dependencies

Story 2.3

#### Acceptance Criteria

1. User should be able to schedule a trip
2. Functional and validation requirements are met

### User Story 2.6: Search Trip Schedules

#### Description

As a Dispatcher, I want search trip schedules so that I can see the status of trips.

Search by Truck Id, driver, start date, end date and status.

UI Flow:

1. User clicks on Search Trips link
2. User is presented with search fields
3. User enters search parameters
4. User sees list of trips based on search criteria
5. All fields from story 2.5 are shown

Search behavior:

1. Search uses AND as a logical operator. All supplied search parameters must hold true in the results.
2. If no search parameters are supplied, all trips are listed
3. Optional: Sorting of results by field

#### Out-of-scope

Date range search

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to search trips based on search criteria

### User Story 2.7: Update Trip Schedule

#### Description

As a Dispatcher, I want to update trip schedules so that can track the status of jobs.

User should be able to update all fields.

UI Flow:

1. From trip search page results, user selects trip to edit.
2. User sees trip form with prepopulated fields
3. User edits fields
4. User submits or cancels form
5. User is brought back to home page

Functional Requirements:

1. Trip ID and Truck ID cannot be changed.

#### Out-of-scope

N/A

#### Dependencies

Story 2.6

#### Acceptance Criteria

1. User is able to update trip details

# Case Study 3: Airline System

## General Description

Airline employees use the Airline System to book passengers on flights.

## User Stories

### User Story 3.1: View Home Page

#### Description

As an Airline Employee I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 3.2: Add Passenger

#### Description

As a Ticket Agent, I want to add a passenger so that I can book that passenger on a flight.

Passenger fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Passenger ID | Numeric | System Generated upon submission |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |

Passengers with the same First name, Last Name, and Birth Date are considered to be the same passenger. The application should not allow duplicate passengers to be entered.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add Passengers

### User Story 3.3: Add Flight

#### Description

As a Flight Dispatcher, I want to add a flight so that the airlines can transport passengers.

Flight fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Flight ID | Numeric | System Generated upon submission |
| Flight Number | String | Yes |
| Origin | String | Yes |
| Destination | String | Yes |
| Departure Date/Time | Date/Time UTC | Yes |
| Arrival Date/Time | Date/Time UTC | Yes |
| Status | 0 – Cancelled  1 – Scheduled  2 – Gate Open  3 – Boarding  4 – Gate Closed  5 – On Time  6 – Delayed  7 – Early  8 – Arrived | Yes |

UI Flow:

1. User clicks on “Add Flight” link.
2. User sees form
3. User submits or cancels form.
4. User receives feedback that flight was added
5. User is brought back to home screen.

Validation:

1. It is possible for a single Flight Number to have multiple legs. For example, flight number PR101 from Manila to Honolulu id the first leg, then the next leg is from Honolulu to San Francisco. Each leg should have its own flight ID.
2. Departure and Arrival times should be in UTC. Therefore, Arrival date/time should always be after Departure date/time.
3. All date/time entered should be after current time.

#### Out-of-scope

1. Conversion to local time of departure and arrival date/time.
2. Schedule conflicts of legs

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to add flight schedules into the system
2. Functional requirements should be met

### User Story 3.4: Search Flights

#### Description

As an Airline Employee, I want to search flights so that I can check on the status of certain flights.

Search by Flight ID, Flight Number, Origin, Destination, Departure/Arrival Date (time optional).

UI Flow:

1. User clicks on “Search Flights”
2. User is presented with search form
3. User enters search parameters and submits
4. User sees list of flights matching all supplied search parameters (AND search)

Functional Requirements

1. AND search must be used. All supplied search parameters must be true for all results.
2. If Flight ID is supplied, ignore other search criteria
3. If no search parameters are supplied, all flights are displayed.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to search flights

### User Story 3.5: Book Passenger on Flight

#### Description

As a Ticket Officer, I want to book a passenger on a flight so that I can generate revenue.

UI Flow:

1. User searches for flight.
2. User selects flight from results.
3. (Optional) User shown passenger search form. User can search by first name, last name, or both.
4. (Optional) User selects passenger from results to add to flight or search again using different parameters
5. User enters First name, middle name, last name of passenger to book
6. User receives feedback that passenger has been added to flight
7. User is brought back to home page after clicking ok.

Validation:

1. Passenger booked must have been added into the system.
2. Passengers cannot be booked twice to the same Flight ID.
3. Passengers cannot be booked to a flight where departure date/time is before current time.

#### Out-of-scope

Checking of schedule conflicts

#### Dependencies

Story 3.4

#### Acceptance Criteria

1. User should be able to book passengers on a flight

### User Story 3.6: Update Flight Details

#### Description

As a Flight Dispatcher, I want update flight details so that others can see the latest status of a flight.

UI Flow:

1. User searches for a flight
2. User selects flight to update
3. User is presented prepopulated form to edit
4. User submits or cancels form
5. User is brought back to home page

Validation:

1. Departure should be before Arrival
2. Unlike Story 3.3, date/time before current date/time is allowed

#### Out-of-scope

N/A

#### Dependencies

Story 3.4

#### Acceptance Criteria

1. User should be able to update flight details

### User Story 3.7: Show Flight Manifest

#### Description

As a Flight Purser I want to see the flight manifest so that I know who the people are on board my flight.

UI Flow:

1. User searches for flight
2. User selects flight
3. User is shown all passengers booked on flight
4. User is returned to Home Page after clicking OK

#### Out-of-scope

(Details here)

#### Dependencies

Story 3.4

#### Acceptance Criteria

1. User is able to get list of passengers booked on a flight

# Case Study 4: Online Shopping System

## General Description

The Online Shopping System is an Ecommerce application that allows shoppers to purchase items online.

## User Stories

### User Story 4.1: View Home Page

#### Description

As a Shopper, I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 4.2: View Products

#### Description

As a Shopper, I want to browse available products so that I can choose which ones to buy.

Products are preloaded into the system. Products must at least have an item name, item description, and price. At least 4 products must be shown on the home screen initially. More products can be shown either by paging or continuous scrolling.

For purposes of prototyping, developer can determine which products to display.

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. Users should be able to see initial products on home page
2. Users should be able to see more products by paging or scrolling

### User Story 4.3: Add Payment Method

#### Description

As a Shopper, I want to add a payment method so that I don’t have to reenter payment method details for every purchase.

Payment Method fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Payment Method ID | Numeric | Yes, System Generated |
| Type | Card, PayPal | Yes |
| Card Number | String | Yes, if Card |
| Expiration Date | String | Yes, if Card |
| Short Name | String | Yes |
| PayPal ID | String | Yes, if PayPal |
|  |  |  |

UI Flow:

1. User Clicks on “Account Maintenance”
2. User clicks on “Add Payment method”
3. User sees payment method form
4. User cancels or submits form
5. User is brought back to home page

Validation:

1. Short Name should be unique
2. Expiration date on Card should be after current date.
3. User should only supply month and year of expiration date. Expiration is always at end of month
4. Card number should have 16 digits

#### Out-of-scope

1. Checksum validation of credit card number
2. PayPal ID verification

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to add payment methods

### User Story 4.4: Remove Payment Method

#### Description

As a Shopper, I want to delete a payment method so that a payment method can’t be used for shopping.

UI Flow:

1. User clicks on “Account Maintenance”
2. User is shown Short Name list of payment methods
3. User clicks on payment method to delete
4. User receives feedback that payment method is deleted
5. User is shown updated list of payment methods
6. User clicks on link to return to home page

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to delete payment method

### User Story 4.5: Add Products to Shopping Cart

#### Description

As a Shopper, I want to add products to the shopping cart so that I can buy products on checkout.

UI Flow:

1. User clicks on product to purchase
2. User is presented product screen
3. User can choose amount to purchase
4. User clicks to add product to shopping cart
5. User can continue shopping or view shopping cart
6. If user views shopping cart, shopping cart should show all items in cart, unit price, amount, total price per item, and total amount.

Validation:

1. Item amount cannot be zero.
2. User can add the same item more than once. This is to support multiple variants in future stories.

Other Details:

1. Optional: Persist shopping cart state to back-end

#### Out-of-scope

N/A

#### Dependencies

Story 4.2

#### Acceptance Criteria

1. User should be able to add products to a cart

### User Story 4.6: Checkout Shopping Cart

#### Description

As a Shopper, I want to checkout a shopping cart so that I can complete my purchase.

Shopper should be able to choose payment method on checkout. Order and all details should be saved to database when checkout is done. Shopping cart should be emptied after checkout.

Optional: Add payment method on checkout.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to checkout shopping cart to complete order

### User Story 4.7: List Pending Orders

#### Description

As a Shopper, I want to view my pending orders so that I will know when my purchases will be delivered.

UI Flow:

1. User clicks on “My Orders”
2. User sees list of dates with total order price.
3. User selects order from list
4. User sees order screen with all order details.

#### Out-of-scope

Order status (Processed, In Transit, Delivered)

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to see past orders

### User Story 4.8: View/Modify Contents of Shopping Cart

#### Description

As a Shopper, I want to modify shopping cart contents so that I can change what I want to buy.

UI Flow:

1. User clicks on shopping cart to view contents
2. User can change item quantity for items
3. User can delete item from cart
4. Cart should update total amounts after each action
5. User can continue shopping or checkout

Validation:

1. Item amount cannot be reduced to zero. Item must be deleted in this case.

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. User can change item amounts in cart
2. User can delete items in cart
3. User can see updates to totals after modifying cart

# Case Study 5: Credit Card System

## General Description

The system is to be used be a credit card holder to manage existing and apply for new credit cards.

## User Stories

### User Story 5.1: View Home Page

#### Description

As a User I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 5.2: Modify User Profile

#### Description

As a Credit Card User, I want to update my user profile so that I can update my billing details.

User profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Home Address | String | Yes |
| Office Address | String | No |
| Phone Number | String | Yes |
| Monthly Income | Numeric | Yes |
|  |  |  |

UI Flow:

1. User clicks on Account Maintenance
2. User sees prepopulated form
3. User edits form fields
4. User either cancels or submits changes

Validation:

1. Birth date cannot be after 18 years before current date. (User must be at least 18 years old)

#### Out-of-scope

Support for multiple users is out of scope for this prototype.

#### Dependencies

Initial profile should already be in the database

#### Acceptance Criteria

1. User can modify profile

### User Story 5.3: Add Supplementary User

#### Description

As a Credit Card User, I want to add a person as a supplementary user so that I can apply for a supplementary card.

User profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| ID | Numeric | System Generated |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Home Address | String | Yes |
| Office Address | String | No |
| Phone Number | String | Yes |
| Relationship | String | Yes |

Validation:

1. Birth Date cannot be after 15 years before current date (Must be at least 15 years old)

#### Out-of-scope

Relationship validation (son older than parent)

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to add supplementary users to the system.

### User Story 5.4: Apply for a Credit Card

#### Description

As a Credit Card User, I want to apply for a credit card so that I can get a credit card.

Credit card types are Bronze, Silver, Gold, Platinum with initial credit limits of P25K, P50K, P100K, P250K respectively. User already has Bronze Card. User gets approved if credit limit of card applied for is less than or equal to 2 \* monthly salary.

Credit Card fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| ID | Numeric | System Generated if approved |
| Card Number | String (####-####-####-####) | System Generated if approved |
| Type | Bronze, Silver, Gold, Platinum | Yes. If Supplementary, same type as parent |
| Name on Card | String | Yes |
| Supplementary | Boolean | Yes |
| Parent Card ID | Numeric | Yes if Supplementary |
| Billing Address | String | Yes if parent card |
| Credit Limit | Numeric | Yes |
| Expiration Date | Date | System Generated if approved, 4 years from current date |
| Supplementary User | String on form. Recommended foreign key in back-end | From Supplementary Users if Supplementary |
| Active | Boolean | Yes |

UI Flow:

1. User clicks in “Apply for New Card”
2. User fills out application form
3. User submits application
4. User sees result of application (Approved or Denied message)

Validation:

1. Credit limit defaults to corresponding values by type above but can be changed to lower amount. Cannot be lower than P10K
2. Fields for supplementary card should be present, but non-functional for this story

#### Out-of-scope

1. Checksum valid credit card number
2. Supplementary card applications

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to apply for a credit card
2. User should be able to have application approved/disapproved

### User Story 5.5: Apply for a Supplementary Credit Card

#### Description

As a Credit Card User, I want to apply for a supplementary credit card for a registered supplementary user so that I can give credit cards to my dependents.

Story is the same as Story 5.4 but with the following differences:

1. User selects parent card from list of active credit cards.
2. Supplementary card will be same type as parent card
3. User adds fills in supplementary user field with supplementary user name (first name, middle name last name)
4. Card automatically approved up to a maximum of 4 supplementary cards per parent credit card.
5. Supplementary cards cannot have supplementary cards.

Validation:

1. Supplementary user must have been entered into the system.

#### Out-of-scope

N/A

#### Dependencies

Story 5.4

#### Acceptance Criteria

1. User should be able to apply for supplementary credit cards

### User Story 5.6: List Credit Cards

#### Description

As a Credit Card User, I want to see a list of my existing credit cards so that I can manage my expenses and credit lines.

UI Flow:

1. User Clicks on “My Credit Cards”
2. User sees list of credit cards numbers, current balance and credit limit

Balance should be based on transactions already in the database. Transactions are assumed to be entered into the database by a separate system. For purposes of prototyping, developer should enter dummy transactions in database.

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| ID | Numeric | Yes |
| Card Number | String (####-####-####-####) | Yes |
| Vendor | String | Yes |
| Amount | Numeric | Yes |
| Date | Date | Yes |

Functional Requirements:

1. Credit card balance is defined as total amount charged to the card that is not yet paid.
2. Listed Cards should show balance and credit limit.
3. It is possible for amount to be negative in the case of refunds, cashback, or payments
4. User paying off balance should be “Payment” as vendor and negative amount.
5. Transactions charged to supplementary cards should be included in calculating the balance of their principal card.

#### Out-of-scope

1. Viewing of individual transactions is in Story 5.8
2. Balance of supplementary cards

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to see current balance of owned principal credit cards

### User Story 5.7: View/Update Credit Card Details

#### Description

As a Credit Card User, I want to view and update my credit card details so that I can manage my card.

Applies to parent cards. User can only change billing address active/inactive. Deactivation of parent card deactivates all supplementary cards.

UI Flow:

1. User Clicks on “My Credit Cards”
2. User sees list of credit cards numbers, current balance and credit limit (Story 5.6)
3. User clicks on a card to edit
4. User is shown card form with editable fields prepopulated. Non-editable fields should just be displayed and not in editable form.
5. User cancels or submits edited form

#### Out-of-scope

N/A

#### Dependencies

Story 5.6

#### Acceptance Criteria

1. User should be able to edit card details.

### User Story 5.8: View Transactions for Credit Card

#### Description

As a Credit Card User, I want to view all transactions for my credit card so that I can manage my expenses.

UI Flow:

1. User Clicks on “My Credit Cards”
2. User sees list of credit cards numbers, current balance and credit limit (Story 5.6)
3. User clicks on a card to view
4. User sees all transactions charged to that card

Functional Requirements:

1. Unlike Story 5.6, principal cards should not show transactions charged to supplementary cards

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. User should be able to view transactions charged to a card

# Case Study 6: Online Banking

## General Description

System is used by bank depositors to manage their accounts online.

GENERAL NOTE: Database transactions should be implemented where appropriate.

## User Stories

### User Story 6.1: View Home Page

#### Description

As a Depositor I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 6.2: List Accounts on Home Page

#### Description

As a Depositor, I want to see an overview of my accounts so that I can quickly see my accounts.

Account overview should be immediately viewable from home page. Accounts should show Account number, short name, type (Savings, Checking), and current balance. Current balance should be based on existing transactions.

For purposes of prototype, existing account should already be in place. Also, sample transactions should already be in the database.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to see account overview on home page.

### User Story 6.3: Open an Account

#### Description

As a Depositor, I want to open an account so that I can deposit money.

UI Flow:

1. User clicks on “Open New Account”
2. User selects type of account (Savings, Checking)
3. User selects existing account to transfer money from.
4. User receives feedback that account has been opened
5. User is brought back to account overview

Functional Requirements and Validation:

1. Minimum balance for opening an account is P500.
2. Amount transferred cannot be greater than source account balance.
3. Transactions for withdrawing from source account and depositing to new account should be recorded in the database.

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. User should be able to open an account

### User Story 6.4: Modify Account Details

#### Description

As a Depositor I want to modify account short name so that I can manage my accounts.

UI Flow:

1. User select account in account overview page.
2. User is shown form to change short name
3. User cancels or submits form
4. User is brought back to home page showing updated short name

#### Out-of-scope

N/A

#### Dependencies

Story 6.2

#### Acceptance Criteria

1. User can modify account short name

### User Story 6.5: Transfer Money Between Accounts

#### Description

As a Depositor, I want to transfer money between my accounts so that I can manage my account balances.

UI Flow:

1. User clicks on “Transfer Funds”
2. User selects account to transfer from
3. User selects amount to transfer
4. User selects account to transfer to
5. User receives feedback that transfer was successful
6. User is brought back to home page with updated account balances

Functional Requirements and Validation:

1. User cannot transfer to same account
2. Amount to transfer must be greater than zero
3. Amount cannot be greater than source account balance
4. When selecting account, user should see account number and short name
5. When entering amount, source account balance should be displayed
6. Transactions should be saved in database

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to transfer money between accounts

### User Story 6.6: Add Merchant

#### Description

As a Depositor, I want to add a merchant so that I can pay my bills online.

UI Flow:

1. User clicks on “Add Merchant”
2. User sees form and enters Merchant name
3. User cancels or submits Merchant
4. User is brought back to home page

Validation:

1. Merchant name is free form and not blank
2. Cannot add same merchant

#### Out-of-scope

1. Smart checking of merchant name
2. UID or unique code for merchant

#### Dependencies

N/A

#### Acceptance Criteria

1. User can add merchant

### User Story 6.7: Pay Merchant

#### Description

As a Depositor, I want to pay a merchant from my account balance so that I can pay my bills online.

UI Flow:

1. User clicks on “Make Payment”
2. User selects merchant
3. User selects account to pay from
4. User selects amount to pay
5. User receives feedback that merchant is paid
6. User is brought back to home page with updated balance

Validation:

1. Amount should be greater than zero
2. Amount should be less than or equal to account balance
3. Merchant should be selectable from drop down or some other similar widget

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to pay merchants

### User Story 6.8: List Transactions for an Account

#### Description

As a Depositor, I want to see all transactions for an account so that I can see account activity.

UI Flow:

1. User selects account from home page
2. User sees list of transactions for that account

Optional:

1. Support for paging
2. Support for date range

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. User can see all transaction for a selected account

# Case Study 7: College Enrollment System

## General Description

The system is to be used by a school’s registrar to register students and enroll them in courses

## User Stories

### User Story 7.1: View Home Page

#### Description

As a Registrar I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 7.2: Add Student

#### Description

As a Registrar I want to save student information so that I can enroll students.

Student profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Student ID | Numeric | System Generated after submission |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Course | String | Yes |

UI Flow:

1. User clicks on “Add Student”
2. User fills out form
3. User submits or cancels form
4. User receives feedback that student has been added
5. User is returned to home page

Functional Requirements and Validation:

1. User cannot add the same student twice. The system should treat two students as identical if they share the same first name, last name, and birth date.
2. Birth date should not be after current date

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add students into the system

### User Story 7.3: List Students

#### Description

As a Registrar I want to see a list of students enrolled so that I can manage their enrollments.

UI Flow:

1. User clicks on “List Students”
2. User sees list of students with Student ID, Full name, course

Optional:

1. Paging of results

#### Out-of-scope

N/A

#### Dependencies

N/A

#### Acceptance Criteria

1. User can bring up complete list of students

### User Story 7.4: Search for Classes

#### Description

As a Registrar I want to search for classes so that I can choose which classes to enroll students in.

Class fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Class ID | Numeric | Yes, System Generated |
| Course Code | String | Yes |
| Course Name | String | Yes |
| Schedule | String | Yes |
| Instructor | String | Yes |
| Location | String | Yes |
| Units | Numeric | Yes |
| Class Size | Numeric | Yes |
|  |  |  |

Searchable fields are Course Code, Course name, Schedule, Instructor

Ui Flow:

1. User clicks on Search Classes
2. User enters search criteria
3. User sees search results

Functional Requirements:

1. If no search criteria entered, list all classes
2. Search is AND search
3. Search results should display Course code, course name, schedule, instructor

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to search classes using search criteria.

### User Story 7.5: View Class Details

#### Description

As a Registrar I want to view class details so that I can decide on enrolling students for that class.

UI Flow:

1. User clicks on “Search Classes”
2. User selects class from search results.
3. User sees details of selected class

Functional Requirements:

1. Details should show class fields from Story 7.4.

#### Out-of-scope

N/A

#### Dependencies

Story 7.4

#### Acceptance Criteria

1. User is able to see class details

### User Story 7.6: Enroll Students in a Class

#### Description

As a Registrar I want to enroll a student in a class so that students can fulfill their course requirements.

UI Flow:

1. User searches or class
2. User selects class to view details
3. User clicks on enroll student
4. User enrolls student (either by entering student id or by selecting student through widget).

Functional Requirements:

1. Students can only have a maximum of 24 units enrolled.
2. A student cannot be enrolled in a full class.
3. A student cannot be enrolled in more than one class of the same course Id.

#### Out-of-scope

N/A

#### Dependencies

Story 7.5

#### Acceptance Criteria

1. User is able to enroll students in a class

### User Story 7.7: List Students Enrolled in a Class

#### Description

As a Registrar I want see who are enrolled in a class so that I can manage class size and student mix.

UI Flow:

1. User searches for class
2. User selects class
3. User sees class details and class list
4. If user enrolls student in class, list is updated.

Functional Requirements:

1. Student list displays Student ID and full name

#### Out-of-scope

Report printing.

#### Dependencies

7.6

#### Acceptance Criteria

1. User can list students enrolled in a class

### User Story 7.8: Unenroll Student from a Class

#### Description

As a Registrar I want to unenroll a student from a class so that I can move that student to another class.

UI Flow:

1. User sees list of students from Story 7.7
2. User selects student to unenroll
3. User sees updated list

#### Out-of-scope

N/A

#### Dependencies

Story 7.7

#### Acceptance Criteria

1. User can unenroll a student from a class.

# Case Study 8: Hotel Booking System

## General Description

The system is to be used by a hotel manager to book guests.

## User Stories

### User Story 8.1: View Home Page

#### Description

As a Manager I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 8.2: Add Guest

#### Description

As a Manager I want to add guest details to the system so that I can improve guest relations.

Guest profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Guest ID | Numeric | Yes, System Generated |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |

UI Flow:

1. User clicks on “Add Guest”
2. User sees new guest form
3. User cancels or submits form
4. User receives feedback that guest was registered
5. User is brought back to home page

Validation:

1. User cannot add same guest more than one. Guest identity is based on first name, last name, birth date. Checking should be case insensitive.
2. Birth date should not be in the future.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can register guests in the system

### User Story 8.3: Search Guests

#### Description

As a Manager I want to search guests so that can check on the guest’s stay at the hotel.

UI Flow:

1. User clicks on Guest Search
2. User enters fist name and/or last name
3. User sees search results

Functional Requirements:

1. If no search criteria is supplied, list all guests
2. Use AND search
3. Results should show Full name, birth date.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can search guests

### User Story 8.4: Search Rooms

#### Description

As a Manager want to search rooms based on type so that I can check for available rooms.

Room fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Room ID | Numeric (or String) | Yes, System Generated |
| Type | String | Yes |
| Price per Night | BigDecimal | Yes |

UI Flow:

1. User clicks on search rooms
2. User selects type
3. User sees rooms of that type

Functional Requirements:

1. User can search for all types of rooms
2. Rooms should already be set in the database

#### Out-of-scope

Different time rates.

#### Dependencies

None.

#### Acceptance Criteria

1. User can search for rooms

### User Story 8.5: Book a Room

#### Description

As a Manager I want to book guests to a room so that the hotel can generate revenue.

Booking Details should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Booking ID | Numeric | Yes, System Generated |
| Guests | List | From Guest Table |
| Room Id | Numeric or String | From Room Table |
| Check-in Date | Date | Yes |
| Check-out Date | Date | Yes |
| Total Price | BigDecimal (Calculated) | Yes |
| Status | Reserved, Confirmed, Canceled | Yes |

UI Flow:

1. User clicks on “Book Room”
2. User sees Book Room form.
3. User enters Room Id (Optional: Choose from drop down or other widget)
4. User enters guest names.
5. User cancels or submit form.
6. User shows booking details with total price.

Functional Requirements:

1. Check-in date cannot be after check-out date.
2. Guest can book more than one room.
3. All guests should be registered in the system.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can book room.

### User Story 8.6: Search Bookings

#### Description

As a Manager I want to search existing bookings that I can manage rooms.

Search for available bookings based on guests.

Ui Flow:

1. User clicks on Search Bookings
2. User enters first name and/or last name of guest
3. User sees all bookings for that guest

Functional Requirements:

1. If no search criteria is supplied, list all bookings
2. Use AND search
3. Results should show Booking ID, Room ID, Check-in check-out dates..

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can search room bookings for a guest

### User Story 8.7: Update Booking

#### Description

As a Manager I want to update a guest’s booking so that I can accommodate changes in a guest’s stay.

Update room and date(s) of stay.

UI Flow:

1. User searches for booking.
2. User sees search results
3. User selects booking
4. User edits booking
5. User cancels or submit booking update
6. User is brought back to Home page

Functional requirements:

1. Only the following fields can be updated
   1. Check-in date if after current date
   2. Check-out date
   3. Guest list (should be at least one guest)
   4. Status
2. Total price should be automatically updated based on date changes
3. Requirements from Story 8.5 still apply

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can edit bookings

# Case Study 9: Bill Consolidation System

## General Description

The Bill Consolidation System is used as a service to combine all of the user’s bills into a single bill. Users can manage which bills will be consolidated.

## User Stories

### User Story 9.1: View Home Page

#### Description

As a Bill Payer I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 9.2: Add a Merchant

#### Description

As a Bill Payer I want a merchant payee so that I can pay bills to that merchant.

UI Flow:

1. User clicks on Add Merchant
2. User is shown form to add Merchant
3. User submits merchant name
4. User receives feedback that new merchant was added
5. User is brought back to Home Page

Validation:

1. Merchant cannot be added more than once
2. Checking of existing merchant name is case insensitive

#### Out-of-scope

Merchant codes

#### Dependencies

N/A

#### Acceptance Criteria

1. User can add merchant payees into the system

### User Story 9.3: Add a Bill

#### Description

As a Bill Payer I want to add a bill so that I can pay the bill online.

Bill fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Bill ID | Numeric | Yes, System Generated upon submission |
| Merchant | String | Yes from merchant table |
| Amount | BigDecimal | Yes |
| Serial no. | String | Yes |
| Bill Date | Date | Yes |
| Date Due | Date | Yes |
|  |  |  |

UI Flow:

1. User clicks on add bill
2. User sees bill form
3. User fills out form
4. User cancels or submits form
5. User receives feedback that billing was entered into the system
6. User is brought back to the home page

Validation:

1. It is possible for amount to be negative, but for purposes of total calculation in other stories, negative amounts should be treated as zero
2. Merchant should come from previously registered merchants in Story 9.2

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to add bills into the system.

### User Story 9.4: Show Total Amount Due for Current Month

#### Description

As a Bill payer I want to see the total amount due of all the bills for the current month so that I know how much to pay for the month.

UI Flow:

1. User immediately see total due for the current month on the home screen

Functional Requirements:

1. Total is the total of all registered bills due for the current month.
2. Date due and not bill date should be used when calculating amount due for the month.
3. If bill has negative amount, it should be treated as zero for total calculation purposes

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User sees amount due for the month
2. Amount is updated whenever bills are registered

### User Story 9.5: Show Total Amount Due for Previous Months

#### Description

As a Bill payer I want to see the total amount due of all the bills for the previous months so that I know how much to pay for the month.

UI Flow:

1. User clicks on Monthly Consolidated History
2. User selects year
3. User sees table of amount due for each month for selected year

Functional Requirements:

1. Total is the total of all registered bill due for the respective month.
2. Date due and not bill date should be used when calculating amount due for the month.
3. If bill has negative amount, it should be treated as zero for total calculation purposes

### User Story 9.6: View Bills from Merchant

#### Description

As a Bill Payer I want to see all bills from a merchant so that I can manage my bills from that merchant.

UI Flow:

1. User clicks on View Merchants
2. User sees list of merchants
3. User selects merchant
4. User sees all bills from that merchant

Functional requirements:

1. Show Serial no., bill date, due date, amount

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User should see all bills from a registered merchant

### User Story 9.7: View History of Monthly Total Amount Due by Merchant

#### Description

As a Bill Payer I want to see bill totals for past months by merchant so that I can see trends in monthly expenses.

UI Flow:

1. User clicks on View Merchants
2. User sees list of merchants
3. User selects merchant
4. User clicks on view billing history
5. User selects year
6. User sees totals for each month for selected year

Functional Requirements:

1. Total is the total of all registered bill due for the respective month.
2. Date due and not bill date should be used when calculating amount due for the month.
3. If bill has negative amount, it should be treated as zero for total calculation purposes

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can see past monthly billing totals by merchant

# Case Study 10: Food Delivery System

## General Description

The application is to be used by a restaurant owner, manager, and or operator. The user receives calls from customers placing orders and enters required information into the system.

## User Stories

### User Story 10.1: View Home Page

#### Description

As a Restaurateur I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 10.2: Add Food Item

#### Description

As a Restaurateur I want to add food menu items so that I can sell that food item to customers.

UI Flow

1. User clicks on “Add Food Item”
2. New screen displays form for adding food item
3. User enters food item details and submits form
4. Once user successfully adds an item, form should be reset so user can add another item.

Food item fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Food Item Name | String | Yes |
| Unit Price | Non-negative numeric | Yes |
| InStock | Boolean | Yes |

Other requirements

1. User cannot add identical food items. All food items should have different names.
2. If user fails to add item due to validation error, application should show specific error.
3. User can go to home screen at any time during the process to exit the feature.

#### Out-of-scope

None

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add food items

### User Story 10.3: List Food Items

#### Description

As a Restaurateur I want to list my food items so that I can manage my menu.

UI Flow

1. User immediately sees food items on the home page.
2. User should be able to toggle visibility of In-stock/out-of-stock/all-items

#### Out-of-scope

None

#### Dependencies

None. Food items can be preloaded for purposes of testing

#### Acceptance Criteria

1. User should see food items entered in the system

### User Story 10.4: Update Food Items

#### Description

As a Restaurateur I want to update food items so that I can modify prices and availability.

UI Flow

1. From list of foods in Story 10.3, user clicks on a food item to edit.
2. New screen appears with update form prepopulated with food item details.
3. User edits food details and submits.
4. On failed validation, application prompts user on errors. Form is not reset and should still retain last entered field values.
5. On successful submission, user is sent back to home screen.

Validation

1. Field requirements from Story 10.2 still apply
2. Food Item Name cannot match any other food item name.

#### Out-of-scope

None

#### Dependencies

Story 10.3

#### Acceptance Criteria

1. User should be able to edit existing food items.

### User Story 10.5: Create an Order

#### Description

As a Restaurateur I want to enter an order so that I can deliver food items.

Order fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Order ID | Numeric | Yes, System Generated upon submission |
| Customer Name | String | Yes |
| Address | String | Yes |
| Contact Number | String | Yes |
| Order Items | List | Yes |
| Total | BigDecimal | Calculated |
| Status | 0 – Received  1 – Kitchen  2 – In Transit  3 – Delivered  4 – Cancelled |  |

Order Item fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Item Name | String | Copied from Food item table |
| Unit price | Numeric | Copied from Food Item table |
| Qty | Positive numeric | Yes |
| Total Item Price | Numeric | Calculated Unit Price \* Qty |

Validation:

1. Food item name and price must come from food items added into the system
2. Cannot add food item if out of stock

#### Out-of-scope

Tax calculations

#### Dependencies

None

#### Acceptance Criteria

1. User is able to place orders into the system

### User Story 10.6: List Orders

#### Description

As a Restaurateur I want to see orders for the day so that I can track orders.

UI Flow:

1. User clicks on “Orders”
2. User sees list of orders
3. User can filter by order status

Functional Requirements:

1. Orders should display Order ID, Customer Name, Address, Status

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can see list of orders

### User Story 10.7: View/Update Order

#### Description

As a Restaurateur I want view and update an order so that I can make changes to the status of an order.

Should be able to change Customer name, address, contact number, status.

UI Flow:

1. User selects order to edit from list of orders
2. User edits order
3. User cancels or submits order update

#### Out-of-scope

Story 10.6

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to update orders

# Case Study 11: Loyalty Rewards System

## General Description

The system is to be used by a cashier at a point-of-sales system to log loyalty points spent or earned by a customer.

## User Stories

### User Story 11.1: View Home Page

#### Description

As a PoS Cashier I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 11.2: Add a Member

#### Description

As a PoS Cashier I want to add new members so that I can increase sales.

Customer profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Customer ID aka card number | UID, String | Yes |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Card expiration date | Date (one year from current date) | Yes |
|  |  |  |

UI Flow:

1. User clicks on “Add Member”
2. User sees form
3. User fills out form
4. User cancels or submits form
5. User receives feedback that member was added
6. User is brought back to home page

Functional Requirements and Validation:

1. Member should be at least 12 years old
2. Member can have multiple cards
3. Customer ID is unique
4. Card expiration is set to default as 1 year after current date, but can be overridden.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add members to loyalty program

### User Story 11.3: Find Members

#### Description

As a PoS Cashier I want to search for members so that I can manage them.

Search by first and/or last name.

UI Flow:

1. User clicks on “Search Members”
2. User supplies search criteria.
3. User sees list of members matching search criteria.

Functional Requirements:

1. Results should show Customer ID and full name
2. If no search criteria is supplied, list all members
3. Search is AND search.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can search members.

### User Story 11.4: Add Reward Points Transaction

#### Description

As a PoS Cashier I want record reward points transactions so that members can spend or earn points.

Transaction fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Transaction ID | Numeric | Yes, System Generated after submission |
| Customer ID | String | Yes (Auto-filled) |
| Transaction Description | String | Yes |
| Amount | Numeric | Yes |
| Transaction Date | Date (default to current date) | Yes |

UI Flow:

1. User searches for member (Story 11.3)
2. User selects member to add transaction
3. User is brought to add transaction form
4. User fills out form
5. User cancels or submits form
6. User is brought back to home page

Functional Requirements and Validation:

1. Cannot spend more points than current balance.
2. Current balance can be calculated as needed or saved as part of member details
3. Customer ID is auto filled based on selected customer
4. Amount is in points, not currency
5. Transaction date defaults to current date but can be overridden.
6. User cannot add transaction to an expired card

#### Out-of-scope

Point calculation based on monetary value of transaction.

#### Dependencies

Story 11.3

#### Acceptance Criteria

1. User can add transactions for members

### User Story 11.5: View Member Transactions

#### Description

As a PoS Cashier I want to view member transactions so that I can see how that member has been using their card.

UI Flow:

1. User searches for member
2. User selects member from results
3. User sees table of all transactions for that member
4. User also sees current point balance

Functional Requirements:

1. Display Transaction ID, Description (can be truncated), Amount and Transaction date
2. List can be sorted by date ascending or descending

#### Out-of-scope

N/A

#### Dependencies

Story 11.3

#### Acceptance Criteria

1. User can view member transactions

### User Story 11.6: View/Edit Member Details

#### Description

As a PoS Cashier I want edit member details so that I can keep member details current.

UI Flow:

1. User searches for member
2. User selects member to edit
3. User cancels or submit member details update
4. User receives feedback that user has been updated
5. User is brought back to home page

Functional requirements:

1. User can only change first name, middle name, last name, birth date.
2. Validation rules for Story 11.2 still apply

#### Out-of-scope

N/A

#### Dependencies

Story 11.3

#### Acceptance Criteria

1. User can edit member

### User Story 11.7: Show Rewards Points Balance in Member Search Results

#### Description

As a PoS Cashier I want to see member balances so that I can inform a member about his points.

UI Flow:

1. User searches for members
2. In addition to previous fields, user can now see balance of members I list of search results

#### Out-of-scope

N/A

#### Dependencies

Story 11.3

#### Acceptance Criteria

1. User sees balance of members in search result

# Case Study 12: Library System

## General Description

The system is used by a librarian to keep track of copies of books that are lent out.

## User Stories

### User Story 12.1: View Home Page

#### Description

As a Librarian I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 12.2: Add a Book

#### Description

As a Librarian I want to add a book to the system so that I can track it.

Book fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Book ID | Numeric | Yes, System Generated upon submission |
| Title | String | Yes |
| Author(s) | String, Comma delimited | Yes |
| ISBN | String | Yes |
| Subject | String, Comma delimited | Yes |
| Location | String | Yes |
| Date published | Date | Yes |
| Date Acquired | Date | Yes |
| Available | Boolean | Yes |

UI Flow:

1. User clicks on “Add Book”
2. User enters book details
3. User cancels or submits book
4. User receives feedback that book was added into the system
5. User is brought back to home page

Functional requirements and Validation:

1. User can add multiple copies of the same title since each copy will be tracked
2. Date Acquired and Date Published cannot be in the future

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can add books into the system

### User Story 12.3: Search for Books

#### Description

As a Librarian I want to search for books so that I can find the book I need.

Search by Title, Author, Subject, ISBN

UI Flow:

1. User clicks on “Search Books”
2. User enters search criteria.
3. User sees list of books matching search criteria

Functional Requirements:

1. Each copy of a book that matches the search criteria should be displayed
2. Searching for Title, Author, Subject should be substring search
3. AND search for Title, Author, Subject
4. If ISBN is supplied, other search criteria are ignored
5. Display Book ID, Title, Author, Subject, ISBN, Available in search results

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can search for books.

### User Story 12.4: View/Edit Book Details

#### Description

As a Librarian I want to edit book details so that I can track book status.

UI Flow:

1. User searches for a book
2. User selects book to view
3. User sees book details
4. User can edit location and availability
5. User cancels or submits book update
6. User receives feedback that book has been updated
7. User is brought back to home page

#### Out-of-scope

N/A

#### Dependencies

Story 12.3

#### Acceptance Criteria

1. User can view and edit book details

### User Story 12.5: Lend a Book

#### Description

As a Librarian I want to record book lending so that I have a record on who borrowed a book and when.

Book Lend fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Book Lend ID | Numeric | Yes, System Generated upon submission |
| Book Id | Numeric | From Book table |
| Borrower Name | String | Yes |
| Date Borrowed | Date | Yes |
| Date Due | Date | Yes |
| Date Returned | Date | No |

UI Flow:

1. User searches for book
2. User selects book to lend
3. User enters lending details
4. User cancels or submits lending details
5. User receives feedback that book lending was registered
6. User is sent back to home page

Functional Requirements and Validation:

1. Cannot lend a book if Available flag is set to false.
2. Cannot lend a book if already lent out.

#### Out-of-scope

N/A

#### Dependencies

Story 12.3

#### Acceptance Criteria

1. User can lend out a book.

### User Story 12.6: Return a Book

#### Description

As a Librarian I want to record a book being returned so that I can lend out the book again.

Returning a book should set Date Returned to current date.

UI Flow:

1. User searches for book
2. User selects book to return.
3. Lending record is updated.
4. User receives feedback that book was returned
5. User is brought back to home page

#### Out-of-scope

N/A

#### Dependencies

Story 12.3

#### Acceptance Criteria

1. User can return a book

### User Story 12.7: View Book Lending History

#### Description

As a Librarian I want to see all the times a book has been borrowed so that I can track a book.

UI Flow:

1. User searches for a book
2. User selects book to view lending history
3. User sees lending history of selected book showing borrower’s name, date borrowed, date returned.

#### Out-of-scope

N/A

#### Dependencies

Story 12.3

#### Acceptance Criteria

1. User can see lending history of a book

# Case Study 13: Expense Tracking System

## General Description

The system is part of a personal financial management program offered by a financial institution client. The system is to be used by the client’s customers to track their spending habits. Customers are expected to make monthly budgets, track their expenses, and check if they have exceeded their budgets.

## User Stories

### User Story 13.1: View Home Page

#### Description

As a User I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 13.2: List Expense Categories

#### Description

As a User I want to see expense categories so that I can categorize my expenses.

Starting expense categories are Utilities, Food, Entertainment. These categories should already be in the system.

From the home page, the user should see an option for “Category Management”. Clicking on this should display a screen with all existing expense categories.

#### Out-of-scope

None

#### Dependencies

None

#### Acceptance Criteria

1. User should be able to see default categories.

### User Story 13.3: Add Expense Category

#### Description

As a User I want to add an expense category so that I can manage my budget better.

#### Functional Requirements

UI Flow

1. From “Category Management” screen in Story 13.2, there should be a text field and a button to add a new expense category. User enters a new category name and submits.
2. If user successfully adds a new category, Category Management screen should immediately refresh and show updated list of categories, otherwise, an error message should appear showing the specific error.

Validation

1. A category cannot be added if it already exists. Category name checking should be case-insensitive.

#### Out-of-scope

None

#### Dependencies

Story 13.4

#### Acceptance Criteria

1. User should be able to add a category through the category management screen

### User Story 13.4: Set Budget for a Category by Month

#### Description

As a User I want to be able to set the monthly budget for a category so that I can budget my money.

The user should be able to set the budget for any category for current or future months. The budget amount is not a recurring value, meaning that each month can have a different value. For example, budget for food might be higher for December or budget for utilities might be higher for summer months.

#### Functional Requirement

UI Flow

1. User goes to separate budget management screen.
2. User selects expense category
3. User enters month and year
4. User enters budget amount
5. User submits budget and is prompted if successful.

UI Requirements

1. User should be able to select expense category without having to type category name.
2. Month and year should default to current date.
3. If budget already exists, user should be asked if they want to update existing budget amount.

Validation

1. User cannot update budgets for past months.
2. Budget amount should be greater than or equal to zero.

#### Out-of-scope

None

#### Dependencies

Story 13.2

#### Acceptance Criteria

1. User is be able to create new budget for a month.
2. User can update existing budget amounts for current or future months.

### User Story 13.5: Add Expense

#### Description

As a User I want to add an expense transaction under a budget category so that I can track my expenses under that category.

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Expense ID | Numeric | System generated after submission |
| Category | String | Yes, from categories |
| Description | String | Yes |
| Amount | BigDecimal | Yes |
| Date | Date | Yes |
|  |  |  |

UI Flow:

1. User clicks on “Add Expense”
2. User fills out expense form
3. User cancels or submits form
4. User is brought back to home page

Validation:

1. Amount must be greater than zero.
2. Date cannot be future date.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can add an expense

### User Story 13.6: Show Budget Report for Current Month

#### Description

As a User I want to see a budget report for the current month so that I can see how well I’m budgeting my money.

UI Flow:

1. Default home page view user sees total amount for all expenses for the current month, total budget, and remaining budget for the current month.

Functional Requirements:

1. Amounts displayed should be updated when changed by other stories.
2. User should also see if they are over budget.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User immediately sees budget status for current month on home page

### User Story 13.7: Show Budget Report for a Category

#### Description

As a User I want to see all my expenses for an expense category so that I can see how much I’m spending for that category.

UI Flow:

1. From Home Page budget report, user click on “Select category to display”
2. User selects category
3. User sees all expenses for that category for the current month, total amount for expenses, budget for the category for current month, and difference.

#### Out-of-scope

N/A

#### Dependencies

Story 13.6

#### Acceptance Criteria

1. User can see budget report for a category for the current month

### User Story 13.8: Show Category Expense Totals for a Given Month/Year

#### Description

As a User I want to see total balance for a selected category for a selected month/year so that I can see how much I’ve spent in the past.

This is a continuation of Story 13.7

UI Flow:

1. After viewing category expense report for current month, user chooses previous month/year to display expense report.
2. User sees expense report for selected category for selected month/year.
3. User can choose another month/year to view.

Functional Requirements:

1. If no budget setting is found, assume budget of 0.

#### Out-of-scope

N/A

#### Dependencies

Story 13.7

#### Acceptance Criteria

1. User can view historical budget reports for selected categories

# Case Study 14: Car Rental System

## General Description

The system is used by the operator of a car rental system to keep track of cars and rentals.

## User Stories

User Story 14.1: View Home Page

*Description*

*As a Manager I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.*

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

*Out-of-scope*

Actual content of reserved areas is out of scope and will be specified by other stories.

*Dependencies*

None

*Acceptance Criteria*

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 14.2: Add a Car

#### Description

As a Manager I want add a car so that I can rent it out to customers.

Car fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Car ID | Numeric | Yes, System Generated upon submission |
| Model | String | Yes |
| Year | Numeric | Yes |
| Color | String | Yes |
| License Plate Number | String | Yes |
| Date Acquired | Date | Yes |
| Seating Capacity | Numeric | Yes |

UI Flow:

1. User clicks on “Add Car”
2. User is shown new car form.
3. User cancels or submit form
4. User is brought back to home screen

Functional Requirements and Validation:

1. License Plate Number cannot be the same as another car’s
2. Seating capacity must be greater than or equal to 2

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can add cars into the system

### User Story 14.3: Search Cars

#### Description

As a Manger I want to search existing cars so that I can find one to rent out.

UI Flow:

1. User clicks on “Search Cars”
2. User enters search criteria (id, model and/or seating capacity).
3. User sees search results showing id, model, color, seating capacity, license number

Functional Requirements:

1. If no search criteria supplied, all cars are listed
2. If id is supplied, ignore other search criteria
3. If model and/or seating capacity are supply, AND search is used.

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can search for cars

### User Story 14.4: View/Edit Car Details

#### Description

As a Manager I want to edit car details so that I can update status of my cars.

UI Flow:

1. User searches for a car
2. User selects car from search results
3. User can change car color, license plate, and/or seating capacity
4. User submits update
5. User is brought back to home screen

Validation:

1. Field validation rules from Story 14.2 will apply

#### Out-of-scope

N/A

#### Dependencies

Story 14.3

#### Acceptance Criteria

1. User can edit car details

### User Story 14.5: Add Rental

#### Description

As a Manager I want to enter car rentals so that I can keep track of my cars.

Rental fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Rental ID | Numeric | Yes, System Generated |
| Customer Name | String | Yes |
| Date Rented | Date | Yes |
| Date Due | Date | Yes |
| Date Returned | Date | No |
| Car ID | Numeric | From Car Table |
|  |  |  |

UI Flow:

1. User searches for cars
2. User selects car to view from Story 14.4
3. User clicks on “Rent”
4. User enters rental details.

Functional Requirements:

1. Car cannot be rented out if car is currently out for rent.
2. Date rented should be before date due
3. Date returned should blank initially

#### Out-of-scope

N/A

#### Dependencies

None

#### Acceptance Criteria

1. User can rent out a car

### User Story 14.6: Search Rentals

#### Description

As a Manager I want to search rentals so I can check on car status.

Search by car ID, dates, and which cars are overdue.

UI Flow:

1. User clicks on “Search Rentals”
2. User enters search criteria based on car ID, date rented, date due, date returned
3. User sees search results showing car id, date rented, date due, date returned
4. User can toggle display of rentals which are overdue (past due date and not returned)

#### Out-of-scope

Date ranges

#### Dependencies

None

#### Acceptance Criteria

1. User can search rentals
2. User can toggle visibility of overdue rentals

### User Story 14.7: Update Rental

#### Description

As a Manager I want to update rentals so that I can record changes in car rental status.

Story is similar to 14.5.

UI Flow:

1. User searches for cars
2. User selects car to view from Story 14.4
3. Instead of “Rent” link, user sees and clicks on “Update Rental”
4. User sees current rental details
5. User updates rental details.

Functional Requirements:

1. User can update customer name and any date field
2. Date returned should be after date rented and cannot be in the future

#### Out-of-scope

N/A

#### Dependencies

Story 14.5

#### Acceptance Criteria

1. User can update car rentals

# Case Study 15: Developer Skills System

## General Description

The system is used by HR managers to manage a roster of developers and their technical skills. It allows HR to look for developers with specific skillsets.

## User Stories

### User Story 15.1: View Home Page

#### Description

As an HR Manager I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story 15.2: Add Skill

#### Description

As an HR Manager I want to add a skill so that I can search people with that skill later.

UI Flow:

1. User clicks on “Skills Management -> Add Skill”
2. User adds new skill.
3. User receives feedback that skill was added.

Functional Requirements and Validation:

1. Identically named skills cannot be added regardless of case. For example, “Rest Services” and “REST Services” are considered identical.
2. Skill names are free form

#### Out-of-scope

Semantic identification of skills. It is not necessary for this story to identify that “Java EE” and “Java Enterprise Edition” are identical skills.

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add a skill to the system.

### User Story 15.3: Add Developer

#### Description

As an HR Manager I want to add a developer so that I can track his skill set.

Developer profile fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Employee ID | String | Yes |
| First Name | String | Yes |
| Middle Name | String | No |
| Last Name | String | Yes |
| Birth Date | Date | Yes |
| Position | String | Yes |

UI Flow:

1. User clicks on “Add Developer”
2. User sees form
3. User cancels or submits form
4. On submit, user receives feedback that developer was added. User is given option to add another developer
5. User is brought back to home page when done with adding developers

Functional Requirements and Validation:

1. It is assumed user will retrieve developer’s employee id from an external system. User will enter the employee ID when adding developers.
2. Employee ID is the unique identifier for developers. No two developers can have the same employee ID
3. Birth date cannot be after current date
4. Position is free form

#### Out-of-scope

Automated generation of Employee ID

#### Dependencies

None

#### Acceptance Criteria

1. User is able to add developers

### User Story 15.4: Search Developers

#### Description

As an HR Manager I want to search for developers with specific skills so that I can fulfill project talent requirements.

Search by skill, skill level, first name, last name, minimum months of experience. This is an AND search for all search parameters supplied by the user.

UI Flow:

1. User clicks on “Search Developers”
2. User is presented with search fields
3. User submits search parameters
4. User sees search results
5. User can do another search or go to homepage

Functional Requirements and Validation:

1. If no search parameters are supplied, all developers are displayed
2. Paged search is optional
3. Wildcards and partial matching of search criteria is optional

Implementation Notes:

1. If Story 15.5 has not yet been implemented, developers should have skill assessments associated with them in the database to test this story with skills.
2. Search results must be generated from the back end. Doing the search from the front end will not scale for thousands of developers.

#### Out-of-scope

Multiple skill search

#### Dependencies

Functionality of some search criteria is dependent on Story 15.5

#### Acceptance Criteria

1. User is able to search for developers

### User Story 15.5: Add Skill Assessment

#### Description

As an HR Manager I want to add skills and corresponding skill level to a developer so that I can find developers with that skill.

Skill Assessment fields should include the following:

|  |  |  |
| --- | --- | --- |
| Field | Type | Required? |
| Skill Name | From Skills | Yes, from Skills |
| Months Experience | Numeric | Yes |
| Skill Level | Numeric Preset: 0 – None 1 – Basic 2 – Foundation  3 – Experienced  4 – Expert  5 – Thought Leader | Yes |
| Employee ID | String | Yes, from Developer table |

UI Flow:

1. User searches for developer.
2. From search results, user selects developer
3. User is shown developer details screen
4. User clicks on add skill assessment
5. User fills out skill assessment
6. User submits or cancels skill assessment
7. User is brought back to developer details screen with updated skills
8. User can return to home page or add another skill

Functional Requirements and Validation:

1. Developer can only have one assessment per skill. User cannot add a skill assessment if developer already has an assessment for that skill. Assessment updates is covered by Story15.6
2. Skill Name should be presented as a drop-down list or some other method that prevents invalid input
3. Skill Level should be presented to the user through a drop-down list, radio buttons, or some other method that will prevent invalid input.

#### Out-of-scope

N/A

#### Dependencies

Story 15.4

#### Acceptance Criteria

1. User is able to add skill assessments to a developer

### User Story 15.6: Update Skill Assessment

#### Description

As an HR Manager I want update skill assessments so that I can keep developer skills current.

User can only edit skill level and months of experience

UI Flow:

1. User searches for developer.
2. From search results, user selects developer
3. User is shown developer details screen
4. User clicks on skill assessment to update
5. User edits skill
6. User submits or cancels update
7. User is brought back to developer details screen and sees updated skill

#### Out-of-scope

N/A

#### Dependencies

Story 15.4 & 15.5

#### Acceptance Criteria

1. User is able to update developer skill assessments

### User Story 15.7: Generate Skill Capability Report

#### Description

As a HR Manager I want to generate a report on skill capability so that I can see an overview of the company’s talent pool.

User should be able to generate a report to see the number of developers having a particular skill, broken down by skill level.

Sample Report:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Skill** | **0:None** | **1:Basic** | **2:Foundation** | **3:Experienced** | **4:Expert** | **5:Thought Leader** | **Total** |
| JavaEE | 10 | 20 | 35 | 20 | 10 | 2 | 97 |
| Python | 10 | 5 | 5 | 3 | 2 | 0 | 25 |
| ReactJS | 10 | 10 | 5 | 5 | 3 | 0 | 33 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

UI Flow:

1. User clicks on “Generate Skill Capability Report”
2. User is presented with report.

#### Out-of-scope

1. Filtering by skill or level
2. Conversion to other formats like .xls, .csv, .pdf, etc.

#### Dependencies

None

#### Acceptance Criteria

1. User is able to see skill capability report

# Case Study ##: Application Name

## General Description

(Details here)

## User Stories

### User Story #.1: View Home Page

#### Description

As a (role) I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story #.2: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.3: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.4: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.5: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.6: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.7: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.8: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.9: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.10: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

# Case Study ##: Application Name

## General Description

(Details here)

## User Stories

### User Story #.1: View Home Page

#### Description

As a (role) I want to view the application home page when I log in so that I have easy access to the features of the application and quickly see relevant information.

The user should see the home page of the application after entering the web application’s URL.

The home page should contain the following:

1. Name of application
2. Reserved area for functions. Place holders should be placed instead of actual links for functions. Functions will be added through succeeding user stories.
3. Reserved area for default views. Nothing to be displayed for now. Views to be displayed will be determined by succeeding user stories.

#### Out-of-scope

Actual content of reserved areas is out of scope and will be specified by other stories.

#### Dependencies

None

#### Acceptance Criteria

1. Base application URL leads to home page.
2. Home page displays web application name
3. Home page has visible areas for functions and default views.

### User Story #.2: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.3: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.4: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.5: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.6: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.7: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.8: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.9: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)

### User Story #.10: Short Name

#### Description

As a (role) I want (something) so that (benefit).

(Details of story here)

#### Out-of-scope

(Details here)

#### Dependencies

(Details here)

#### Acceptance Criteria

1. (Criteria 1)
2. (Criteria 2)
3. (Criteria 3)