Software Engineering Project

Requirements Analysis

Project Team:

Matthew Ganpat - 812002121

Devindra Mahadeo - 812117346

Shervonne Cummings - 812000391

Contents

1. Introduction

1.1 Expectations

1.2 Scope

1.3 Definitions

1.4 General Description

1.5 User Characteristics

2. Functional Characteristics

2.1 User Requirements

2.2 System Requirements

3. Non-Functional Requirements

4. Use Cases

5. Data Flow Diagrams

6. Entity Relationship Diagrams

7. Resource Diagrams

8. Sequence Diagrams

9. Activity Diagrams

10. User Interface Screenshots

**1. Introduction**

The purpose of this user requirement document is to specify the requests that the users of the system at Aki’s Bakery are expecting to use when the software is developed in order to develop software according to their needs.

This document is intended to be read by:

* All of the managerial personnel of Aki Bakery.
* All responsible for the management of the developments in question.
* Users which are anyone within the business that has to come into contact with this system that is under development.
* The persons that are actually going to build the system itself, which are the programmers.

1.1 Expectations

* Developing a successful software that can meet the prospect of users and customers that can stand up to the competitors in terms of quality. The user requirements analysis provides direct descriptions of the quality, content and functionality that the user demands, such demands can be categorized as functional requirements and non-functional requirements.

Functional requirements

* These describe the functionality that the system should provide and how these are to be performed within the new system. The functional requirements allows one to understand the tasks that involve the concept of why particular activities are being performed by users, what their limitation and preferences are.

Non-Functional requirements

* The non-functional requirements consists of the different types of users which can be the professionals and personal users, users characteristics description, subjective preferences and the environment in which the system will be used by the user.

**1.2 Scope**

The software implements a web based application that is able to execute relevant business processes when the respective module is chosen. The complexity of this software is hidden from the users and web-app-based-front-end for users to access the system increases the user friendliness and gives an ease when completing difficult or tiresome business processes.

**1.3 List of definitions and abbreviations**

**Task** – specification of application, configuration data, input and or output data files.

**Aki Pro Plus**- the name of the software being developed.

**Web based application**- an application or program that is accessed over a network connection using HTTP, rather than existing within a device’s memory. Web-based applications often run inside a Web browser. However, Web-based applications also may be client-based, where a small part of the program is downloaded to a user’s desktop, but processing is done over the Internet on an external server.

NAT – Network Address Translation.

**1.4 General Description**

**Main Goal**

Aki Pro Plus provides users with a web based system that can be used anywhere once they have access to the internet, which assists with major job processes. This system should be able to assist all its users such as delivery personnel, the bakers, and data entry staff and as well as the executive with their respective job processes.

**General capabilities**

The Aki Pro Plus system is designed to accomplish business processes at Aki’s Bakery through a web-based interface. This web-based system uses resources such as apps, files/information and can be represented on a computer or any mobile devices.

Aki Pro Plus is accessed through the internet by internal users of the business such as delivery truck drivers, sales representatives, office workers as well as external users such as customers.

The system makes sure that no one without the correct credentials can access private information. The manager can set limits to who can have access or update to certain things based on their credentials for example a delivery van driver cannot change his own routes.

Aki Pro Plus will be easy for users to operate.

**1.5 User characteristics**

The following is the listing of the roles that will exist in the Aki Pro Plus system.

* Delivery Van Drivers: The delivery van drivers are responsible for getting orders to customers in a timely manner and recording their transactions.
* Office Workers: The Office staff assigns routes to delivery personnel, update and edit customer and ordering information. They also approve changes to standing orders.
* Customers: The customers place the orders and standing orders in which they wish to purchase from Aki Bakery.
* Sales Representative: This person serves customers, is the bridge between management and customers in respect the pricing and managing sales trends of the products through daily call reports, weekly work plans and monthly analyses.

**2. Functional Requirements**

**2.1 User Requirements**

**User: Delivery van Driver**

1. The van driver shall be able to log into the system using his username and password and automatically retrieve his routes.
2. The van driver shall be able to view a single invoice
3. The van driver shall be able to view the invoices for a customer
4. The van driver shall be able to print an invoice
5. The van driver shall be able to sync his mobile device to the main Aki Pro Plus database, relaying and retrieving changes.
6. The van driver shall be able to access the cash account section of a customer’s invoice.
7. The van driver shall be able to create a new cash account entry for on the spot sales.

**User: Sales representative**

1. The sales representative shall be able to log into the system using their username and password and retrieve their status.
2. The sales representative shall be able to modify standing orders.
3. The sales rep shall be able to generate customers order reports.
4. The sales representative shall be able to generate standing order reports.

**User: Customers**

1. The customer shall be able to make orders through the system.
2. The customer shall be able to edit orders.
3. The customer shall be able to create standing order.
4. The customer shall be able to make changes to standing order.
5. The customer shall be able to log on the system using their credentials.
6. The customer shall be able to access account information.
7. The customer shall be able to access order information.

**User: Office workers**

1. Office workers shall be able to log into the system using their credentials.
2. Office workers shall be able to generate orders for customers that call in to make orders.
3. Office workers shall be able to generate invoices.
4. Office workers shall be able to view the invoices for a customer

**2.2 System Requirements**

1. Aki Pro Plus shall be a web-based user interface software.
2. Aki Pro Plus system shall provide apps through which users can use to interact with system.
3. Aki Pro Plus system should provide a command-line interface through which users can use to interact with system.
4. The language content used in Aki Pro Plus should be English.
5. The system should allow access to any amount of users at the same time.
6. The system shall select the correct resources that are needed to process a task.
7. The system shall only give access to users with specific credential clearance for a specific task based on their job description.
8. The Aki Pro Plus system shall provide the means for users to use a method to idenetify themselves on the system.
9. The Aki Pro Plus system should be able to record the times at which a user accesses the system.
10. The Aki Pro Plus system shall be able to keep information for a very long time period such as standard recipes must be able to be accessed for more than a year.
11. The Aki Pro Plus system must allow persons with high clearance to make changes to the system i.e. IT personnel, maintenance personnel.
12. The Aki Pro Plus system shall be able to work with any time of operating systems, windows, mac, Linux.
13. The Aki Pro Plus system shall provide users with the means to add data files to the system.
14. The Aki Pro Plus system shall provide users with the means to delete data files to the system.
15. The functionality of the sytem should not be restricted when computers of users are behind a firewall or a NAT.
16. All applications in the Aki Pro Plus system should log what they are doing.

**3. Non-Functional requirements**

Authentication – The system must be able to authenticate users within the system It must be able to grant and revoke privileges in order to have a level of control for employees.

Robustness – The system should be able to ensure that the content and content services are available all the time. All system failure must be logged.

Maintainability – The system must be able to be updated easily,

Portability – All users would be able to run system on mobile hardware, since the system is based on the internet there would be no specific requirements regarding portability.

Usability – It would be easy to use, simple for someone with basic computer usage skills to understand for to use it.

User interface- Navigating through features present in the system should be easy.

Availability- This system must be able 24/7 and be extremely reliable.

Deployment- Mobile modules must be able to update through the internet.

Security- Username and password is required every time a user enters the system, or tries to make any changes on the system. A strict execution time would be present; if a user takes too long to complete a task the system would kick them out, having them to reenter their username and password.

Secure transfer of data between the customers and Aki Pro Plus. Sensitive data should be either encrypted over SSL or through HTTPS connections.

System stability - the system must be able to deal with errors in the most efficient manner.

Distribution- new mobile client module versions will be distributed using the internet.

1. The system shall be able to perform its function under stated conditions for a specific period of time.
2. The system shall be able to quickly react to user input.
3. The system should be available for service when requested by end users.
4. The system should be dependable- the ability to deliver service that can be justifiably trusted by users.
5. The system should be scalable- the capability of the system to increase total throughput under an increased load when resources are added.
6. The system should be secure
   * + 1. All system data should be backed up every 2 hours and the backup copies stored in a secure location.
       2. All external communications between the system’s data server and clients shall be encrypted.
7. The system should have a well-formed graphical user interface.
8. The system shall have help facilities.
9. The system shall produce understandable error messages.
10. The system should provide a well-structured user manual.
11. The system should be able maintainable – ability to be changed with new technology or to fix defects within the system

**4. Use Cases**

Customer Account Use Cases

\*Assumption = Customer will have internet access

Use Case #1

|  |  |
| --- | --- |
| Use Case Name | Create new standing order |
| Related Requirement | Must know item names and quantities |
| Goal in context | Successfully creates standing order |
| Pre-Conditions | User must log in and have permissions create standing order |
| Successful end conditions | User successfully creates a standing order |
| Fail end condition | User is unable to create a standing order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flow | 1. User Opens application 2. Opens Standing order module 3. Select create new standing order 4. Enter info 5. Save |

Use Case #3

|  |  |
| --- | --- |
| Use Case Name | Create new order |
| Related Requirement | Must know item names and quantities |
| Goal in context | Successfully creates order |
| Pre-Conditions | User must log in and have permissions create order |
| Successful end conditions | User successfully creates an order |
| Fail end condition | User is unable to create an order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flow | 1. User Opens application 2. Opens order module 3. Select create new order 4. Enter info 5. Save |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Edit standing order |
| Related Requirement | Must know item names, quantities, standing order date |
| Goal in context | Successfully edits standing order |
| Pre-Conditions | User must log in and have permissions edit standing order |
| Successful end conditions | User successfully edits a standing order |
| Fail end condition | User is unable to edits a standing order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flow | 1. User Opens application 2. Opens Standing order module 3. Searches for Standing order 4. Selects standing order 5. Edits standing order 6. Save |
| Evaluation |  |

Use Case #5

|  |  |
| --- | --- |
| Use Case Name | View Accounts information |
| Related Requirement | User must log in |
| Goal in context | User successfully views account information |
| Pre-Conditions | User must log in |
| Successful end conditions | User successfully views account information |
| Fail end condition | User is unable to view account information |
| Primary Actor | Customer |
| Secondary Actor | - |
| Main Flow | 1. User Opens application 2. Selects account information |

Driver Use Cases

Use Case #1

|  |  |
| --- | --- |
| Use Case Name | Driver Login |
| Related Requirement | Must know username and password |
| Goal in context | Successful login |
| Pre-Conditions | Must have login credentials and internet access |
| Successful end conditions | User logs in successfully |
| Fail end condition | User unsuccessful in logging in |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. User Prompted for login credentials 3. User would enter credentials 4. User redirected to main menu |

Use Case #2

|  |  |
| --- | --- |
| Use Case Name | View invoice |
| Related Requirement | Must know Customer Name and date of invoice |
| Goal in context | Successfully view Invoice |
| Pre-Conditions | User must log in and have permissions to view invoices |
| Successful end conditions | User successfully views an invoice |
| Fail end condition | User is unable to view invoices |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flo | 1. User Opens application 2. Opens View invoices module 3. Enters customer name 4. Selects invoice to view 5. Invoice loads and is viewable |

Use Case #3

|  |  |
| --- | --- |
| Use Case Name | Record Cash Purchase |
| Related Requirement | User must know customer name and cash sale amount |
| Goal in context | User successfully records cash purchase |
| Pre-Conditions | User must log in and have permissions to view record cash purchases |
| Successful end conditions | User successfully records a cash purchase |
| Fail end condition | User is unable to record a cash purchase |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Access Invoices 3. Selects add new entry to cash account purchases 4. Enters Cash Purchase info 5. Saves invoice |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Process Returns |
| Related Requirement | User must know monetary amount of returns |
| Goal in context | User successfully processes returns |
| Pre-Conditions | User must log in and have permissions to view record returns |
| Successful end conditions | User successfully processes returns |
| Fail end condition | User is unable to record returns |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Access Invoices 3. Selects add new entry to cash account purchases 4. Enters Cash Purchase info 5. Saves invoice |

Use Case #5

|  |  |
| --- | --- |
| Use Case Name | Print invoices |
| Related Requirement | User must know customer name and date of invoice |
| Goal in context | User successfully prints invoice(s) |
| Pre-Conditions | User must log in and have permissions to view invoices |
| Successful end conditions | User successfully prints invoice(s) |
| Fail end condition | User is unable to print invoice |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Access Invoices via order invoices or cash purchase invoices 3. Selects invoice and opens invoice 4. Selects print invoice |

Use Case #6

|  |  |
| --- | --- |
| Use Case Name | Sync |
| Related Requirement | User must login |
| Goal in context | User successfully syncs device |
| Pre-Conditions | User must have internet access |
| Successful end conditions | User successfully syncs |
| Fail end condition | User is unable sync |
| Primary Actor | Driver |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects sync |

Order Processing Use Cases

Use Case #1

|  |  |
| --- | --- |
| Use Case Name | Create standing order |
| Related Requirement | Must know standing order day, items and quantity |
| Goal in context | Successful creation of standing order |
| Pre-Conditions | Must have customer information and access to create standing orders |
| Successful end conditions | User creates standing order successfully |
| Fail end condition | User unsuccessfully creates standing order |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects Standing order module 3. Select create standing order 4. Enter standing order info 5. Save |
| Evaluation |  |

Use Case #2

|  |  |
| --- | --- |
| Use Case Name | Generate standing order |
| Related Requirement | Must know standing order day, production date |
| Goal in context | Successful generation of standing order |
| Pre-Conditions | Must have a standing order day and a production date |
| Successful end conditions | User generates standing order successfully |
| Fail end condition | User unsuccessfully generates standing order |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects Standing order module 3. Select generate standing order 4. Enter standing order info |
| Evaluation | Service:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/standingorders/generate/198990  Parameters: {"Standingday":"WED" , "productiondate":"11/7/2014"}  Operations: POST operation  Testing Result: Successful |

Use Case #3

|  |  |
| --- | --- |
| Use Case Name | View generated standing order |
| Related Requirement | Must know standing order day, production date |
| Goal in context | Successful viewing of generated standing order |
| Pre-Conditions | Must have customer information and access to create standing orders |
| Successful end conditions | User views standing order successfully |
| Fail end condition | User unsuccessfully generates standing order |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects Standing order module 3. Select generatestanding order 4. Enter standing order info |
| Evaluation | Service:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/orders/WED/198990  Operations: GET operation  Testing Result: Successful |

Use case #2

|  |  |
| --- | --- |
| Use Case Name | Create order |
| Related Requirement | Must know order day, items and quantity |
| Goal in context | Successful creation of order |
| Pre-Conditions | Must have customer information and access to create order |
| Successful end conditions | User creates order successfully |
| Fail end condition | User unsuccessfully creates order |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects order module 3. Select create new order 4. Enter order info 5. Save |
| Evaluation | Service: 64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/standingorder/118809/198990  Operation: GET  Result: Successful |

|  |  |
| --- | --- |
| Use Case Name | Generate a selected order |
| Related Requirement | Must know order day, items and quantity |
| Goal in context | Successful creation of order |
| Pre-Conditions | Must have customer information and access to create order |
| Successful end conditions | User creates order successfully |
| Fail end condition | User unsuccessfully creates order |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects order module 3. Select create new order 4. Enter order info 5. Save |
| Evaluation | Service: 64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/standingorders/generateselested/198990  Operation: POST  Parameters: production date, order number  Result: Successful |

Use Case #3

|  |  |
| --- | --- |
| Use Case Name | Request to modify standing order |
| Related Requirement | Must know customer name, sales trends/reports, item names and quantities |
| Goal in context | Successfully edits standing order |
| Pre-Conditions | User must log in and have permissions edit standing order |
| Successful end conditions | User successfully edits a standing order |
| Fail end condition | User is unable to edits a standing order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flo | 1. User Opens application 2. Opens Standing order module 3. Enters customer to search for 4. Select customer 5. Select standing order for customer 6. Edit info 7. Save |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Generate orders |
| Related Requirement | Must know order days, items and quantity |
| Goal in context | Successful creation of orders |
| Pre-Conditions | Must have customer information and access to create orders |
| Successful end conditions | User creates orders successfully |
| Fail end condition | User unsuccessfully generate orders |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects order module 3. Select generate orders 4. Enter orders’ criteria 5. Generate |
| Evaluation | Service to view orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/orders/WED/198990  Service was successfully tested |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Delete all orders for a day |
| Related Requirement | Must know order days |
| Goal in context | Successful deletion of orders for a day |
| Pre-Conditions | Must have orders already existing |
| Successful end conditions | User deletes orders successfully |
| Fail end condition | User unsuccessfully deletes orders |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects order module 3. Select delete orders 4. Enter orders’ deletion day 5. Deletes orders |
| Evaluation | Service to delete orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/standingorders/deleteorders/198990  Operation: POST  Parameters: standing order day  Result: Successful |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Delete an item |
| Related Requirement | Must know item name |
| Goal in context | Successful deletion of item |
| Pre-Conditions | Must have items already existing |
| Successful end conditions | User deletes item successfully |
| Fail end condition | User unsuccessfully deletes item |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | \*\*\*\*\*\*\*\*\*\*\*   1. User Opens application 2. Selects order module 3. Select delete orders 4. Enter orders’ deletion day 5. Deletes orders |
| Evaluation | Service to delete orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/standingorders/deleteitem/198990  Operation: DELETE  Parameters: standing order day, token  Result: Successful |

|  |  |
| --- | --- |
| Use Case Name | Delete an item from a standing order |
| Related Requirement | Must know item name and order number |
| Goal in context | Successful deletion of item |
| Pre-Conditions | Must have items and standing order already existing |
| Successful end conditions | User deletes item successfully |
| Fail end condition | User unsuccessfully deletes item |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | \*\*\*\*\*\*\*\*\*\*\*   1. User Opens application 2. Selects order module 3. Select delete orders 4. Enter orders’ deletion day 5. Deletes orders |
| Evaluation | Service to delete orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/sitem/ordernumber/itemname/198990  Operation: DELETE  Parameters: ordernum, item name, token  Result: Successful |

Use Case #4

|  |  |
| --- | --- |
| Use Case Name | Retrieve Items for an order |
| Related Requirement | Must know order number |
| Goal in context | Successful retrieval of items on an order |
| Pre-Conditions | Must have orders already existing |
| Successful end conditions | User retrieves items successfully |
| Fail end condition | User unsuccessfully retrieves items |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | 1. User Opens application 2. Selects order module 3. Select items 4. Enter order item |
| Evaluation | Service to delete orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/sitems/119011/198990  Operation: GET  Parameters: Order number (119011 used), token  Result: Successful |

|  |  |
| --- | --- |
| Use Case Name | Delete an item from an order |
| Related Requirement | Must know item name and order number |
| Goal in context | Successful deletion of item |
| Pre-Conditions | Must have items and order already existing |
| Successful end conditions | User deletes item successfully |
| Fail end condition | User unsuccessfully deletes item |
| Primary Actor | Office Worker |
| Secondary Actor | Manager |
| Main Flow | \*\*\*\*\*\*\*\*\*\*\*   1. User Opens application 2. Selects order module 3. Select delete orders 4. Enter orders’ deletion day 5. Deletes orders |
| Evaluation | Service to delete orders:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/sitem/118809/BUNS/198990  Operation: DELETE  Parameters: ordernum (118809), item name (BUNS), token  Result: Successful |

Sales Rep Use Cases

Use Case #1

|  |  |
| --- | --- |
| Use Case Name | Sales Rep Login |
| Related Requirement | Must know username and password |
| Goal in context | Successful login |
| Pre-Conditions | Must have login credentials |
| Successful end conditions | User logs in successfully |
| Fail end condition | User unsuccessful in logging in |
| Primary Actor | Sales Rep |
| Secondary Actor | - |
| Main Flow | 1. User Opens application 2. User Prompted for login credentials 3. User would enter credentials 4. User redirected to main menu |

Use Case #2

|  |  |
| --- | --- |
| Use Case Name | View all items for a standing order |
| Related Requirement | Must know standing ordernumber |
| Goal in context | Successfully views items on standing order |
| Pre-Conditions | User must have permissions to view standing order |
| Successful end conditions | User successfully edits a standing order |
| Fail end condition | User is unable to edits a standing order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flow | 1. User Opens application 2. Opens Standing order module 3. Selects items 4. Enters standing order number |
| Evaluation | Service:  64.28.139.185/akipro/orders.dll/datasnap/rest/TOrderProcess/sitems/118809/198990  Operation: GET Parameters: Standing order number, token |

Use Case #2

|  |  |
| --- | --- |
| Use Case Name | Request to modify standing order |
| Related Requirement | Must know customer name, sales trends/reports, item names and quantities |
| Goal in context | Successfully edits standing order |
| Pre-Conditions | User must log in and have permissions edit standing order |
| Successful end conditions | User successfully edits a standing order |
| Fail end condition | User is unable to edits a standing order |
| Primary Actor | Customer |
| Secondary Actor | Office worker |
| Main Flo | 1. User Opens application 2. Opens Standing order module 3. Enters customer to search for 4. Select customer 5. Select standing order for customer 6. Edit info 7. Save |

Use Case #3

|  |  |
| --- | --- |
| Use Case Name | Generate reports |
| Related Requirement | Must know customer name, item name, sales date range |
| Goal in context | Successfully generates Reports |
| Pre-Conditions | User must log in and have permissions generate reports |
| Successful end conditions | User successfully generates a report |
| Fail end condition | User is unable to generate a report |
| Primary Actor | Sales Rep |
| Secondary Actor | - |
| Main Flo | 1. User Opens application 2. Open Report module 3. Select trend report type 4. Select customer 5. Enter report criteria |
| Evaluation | Service: 64.28.139.185/akipro/reports.dll/datasnap/rest/TReportProcess/report/198990  Operation: PUT  JSON String: {"asatdate": "1/11/2014","bodytext": "","enddate": "20/11/2014","keyvalue": "","numberpara1": "1","numberpara2": "","numberpara3": "","outputformat": "pdf","sendbyemail": "","sendtoaddress": "","startdate": "1/1/2002","stringpara1": "1","stringpara2": "","stringpara3": "","subject": "","token": 53582}  Result: Successful |

|  |  |
| --- | --- |
| Use Case Name | View generated report |
| Related Requirement | Must know report number |
| Goal in context | Successfully generates Reports |
| Pre-Conditions | User must have permissions view reports and must know token of generated report |
| Successful end conditions | User successfully views generated report |
| Fail end condition | User is unable to view generated report |
| Primary Actor | Sales Rep |
| Secondary Actor |  |
| Main Flo | 1. User Opens application 2. Open Report module 3. Select trend report type 4. Select customer 5. Enter report criteria |
| Evaluation | Service:  64.28.139.185/akipro/reports.dll/datasnap/rest/TReportProcess/report/1/53582  Operation: GET  Parameter : token( token must be token of generated report) {"asatdate": "1/11/2014","bodytext": "","enddate": "20/11/2014","keyvalue": "","numberpara1": "1","numberpara2": "","numberpara3": "","outputformat": "pdf","sendbyemail": "","sendtoaddress": "","startdate": "1/1/2002","stringpara1": "1","stringpara2": "","stringpara3": "","subject": "","token": 53582}  Result: Successful |

1. **. Data Flow Diagrams**

Customer Account DFD



Delivery Management DFD



Order Processing DFD



Sales Rep Mobile App DFD



**6. Entity Relationship Diagrams**

Customer ERD



Delivery ERD

Order Processing

Sale Rep ERD



**7. Resource Diagrams**

**Invoice**

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Input Parameter | Output | Description |
| GET | Customer Name/ Invoice Date/Token | Invoice number/ item entries/invoice dates/prices/discounts | Retrieves one invoice |
| POST | Operation/Item name, quantity/customer name | JSON object – with a string to show status of request(successful, failed and why) | Operation = Creating Returns information posted to relevant invoice |
| DELETE | Invoice number/ invoice date/ customer name/token | JSON object – with a string to show status of request(successful, failed and why) | Deletes one chosen invoice |

**Order**

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Input Parameter | Output | Description |
| GET | Customer Name/ Order Date/Token | Order number/ item entries/invoice dates/prices/ discounts | Retrieves one order |
| PUT | Item Name/ Quantity/price/ discount/ token | JSON object – with a string to show status of request (successful, failed and why) | Operation= ‘generate’, ’generate selected’, ‘delete orders’, ‘change date’ |
| POST | Operation/Item name, quantity/customer name/token | JSON object – with a string to show status of request (successful, failed and why) | Operation= ‘generate’, ’generate selected’, ‘delete orders’, ‘change date’ |
| DELETE | Order date/ customer name/token | JSON object – with a string to show status of request | Deletes all orders for a given day |

**Standing Order**

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Input Parameter | Output | Description |
| GET | Customer Name/ Standing Order Date/Token | Standing Order number/ item entries/invoice dates/prices/ discounts | Retrieves one standing order |
| PUT | Item Name/ Quantity / price/ discount/ token | JSON object – with a string to show status of request (successful, failed and why) | Operation= ‘generate’, ’generate selected’, ‘delete orders’, ‘change date’ |
| POST | Operation/Item name, quantity/customer name/token | JSON object – with a string to show status of request (successful, failed and why) | Operation= ‘generate’, ’generate selected’, ‘delete orders’, ‘change date’ |
| DELETE | Standing order date/ customer name/token | JSON object – with a string to show status of request | Deletes all standing orders for a given day |

1. **Sequence Diagrams**

Customer Account Sequence Diagram



Delivery Management Sequence Diagram



Order Processing Sequence Diagram



Sales Rep Sequence Diagram



1. **Activity Diagrams**



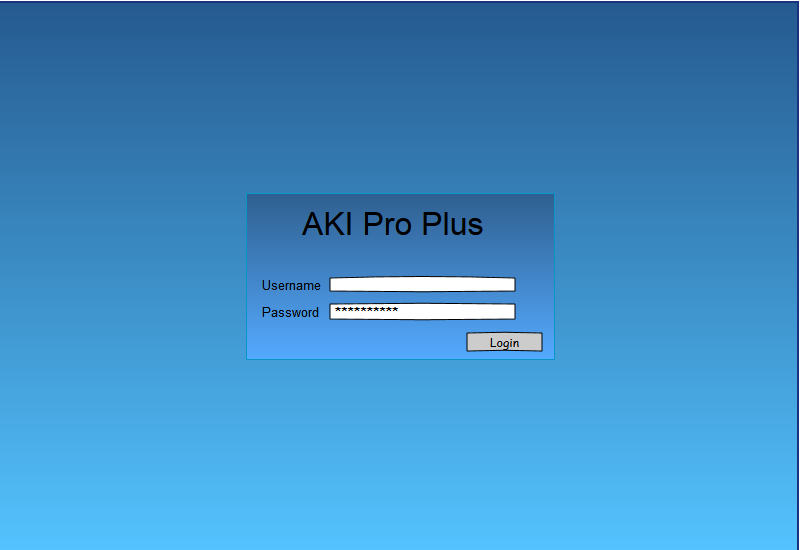




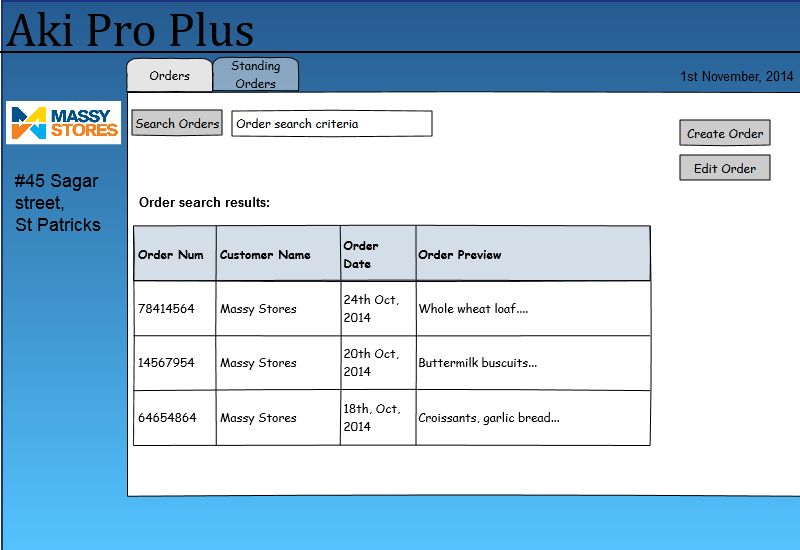
1. **User Interface Design Screenshots**

**Customer**

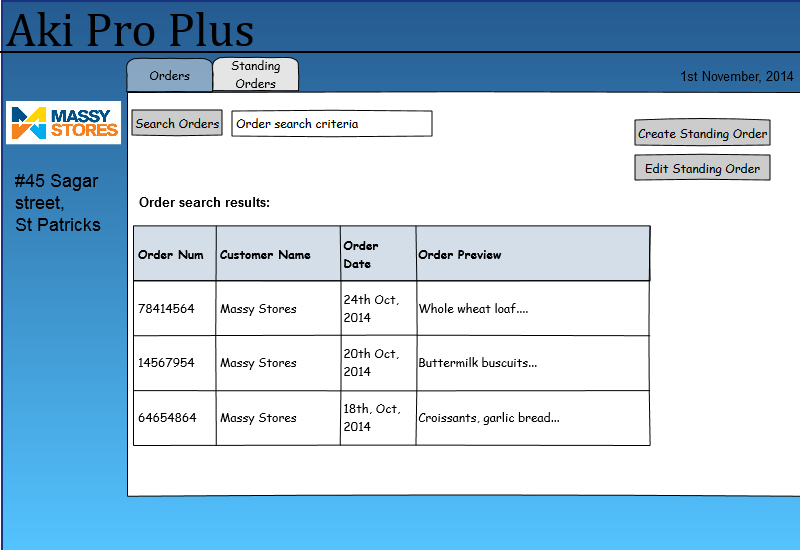
Customer Login Screen

****

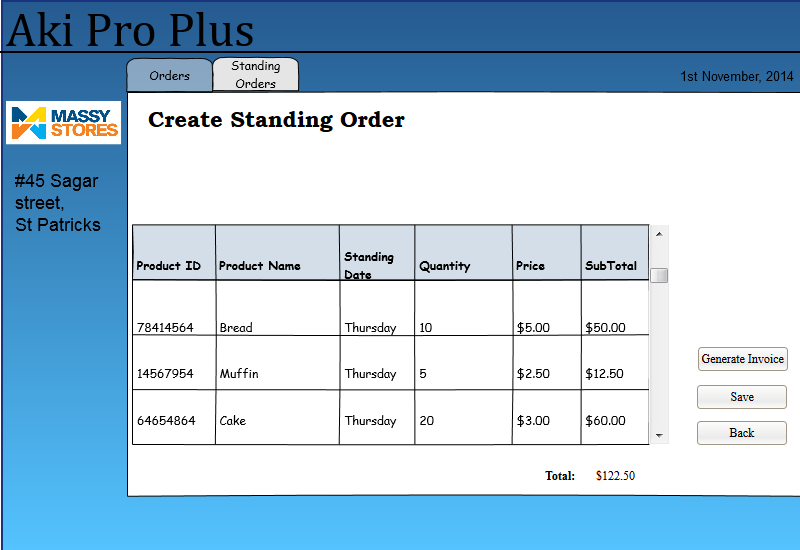
Default page when logged in as Massy Stores. Shows Orders tab with various options.



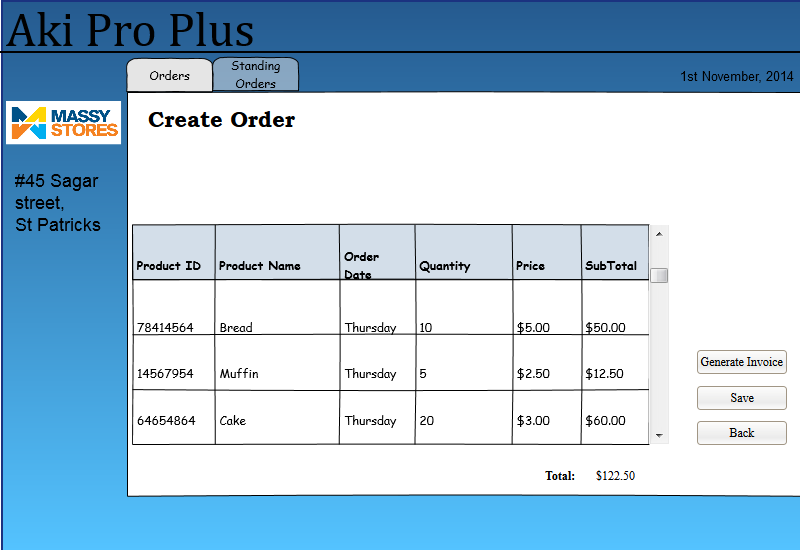
Standing Order tab is shown with various options.



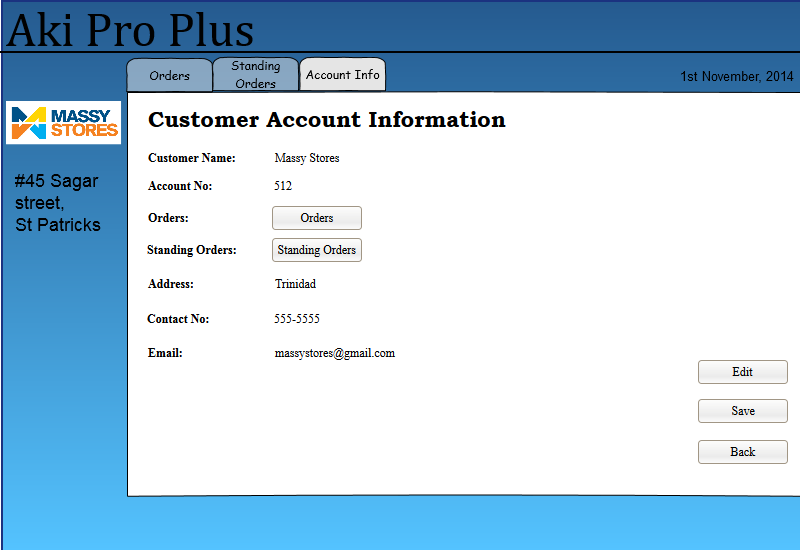
Standing Order tab is shown with various options.



Create Order option is shown with other various options.

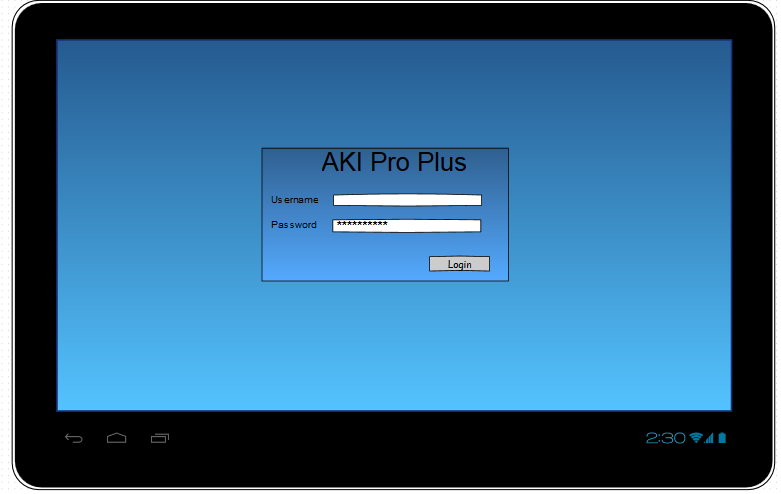


Customer Account option is shown with other various options.

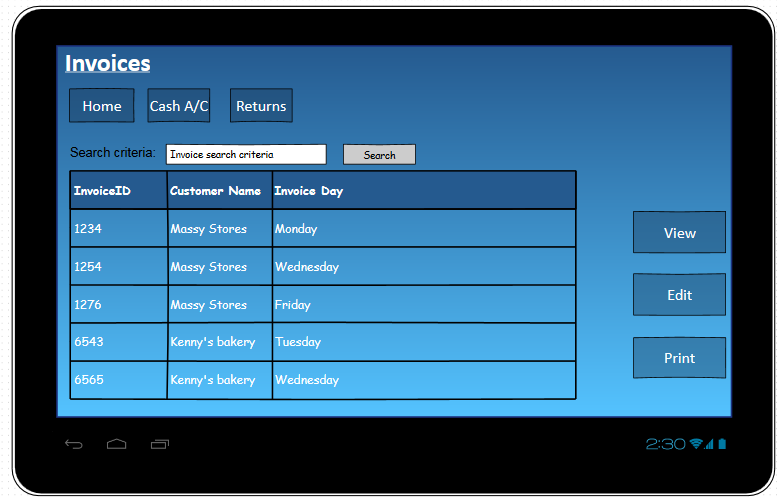


**Driver**

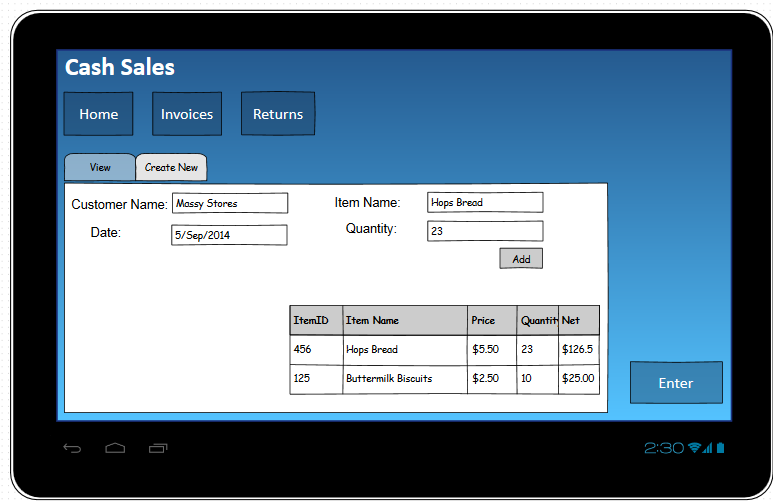
Driver log in



Driver Home Page

Driver Invoices  


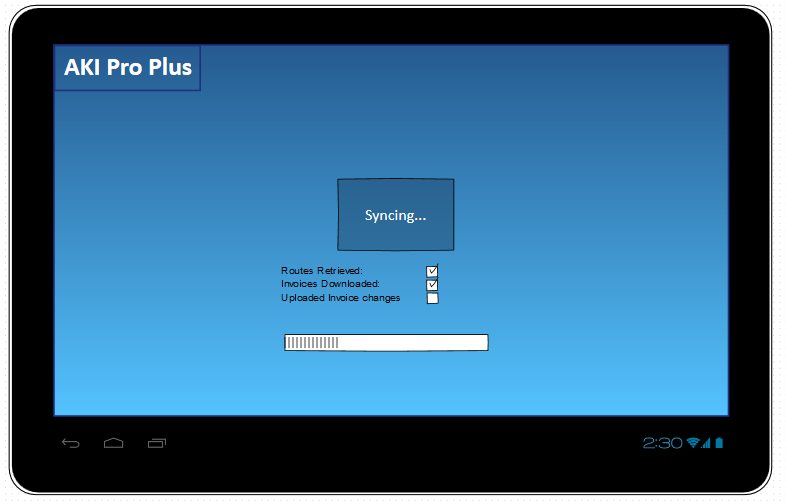
Driver Cash sales



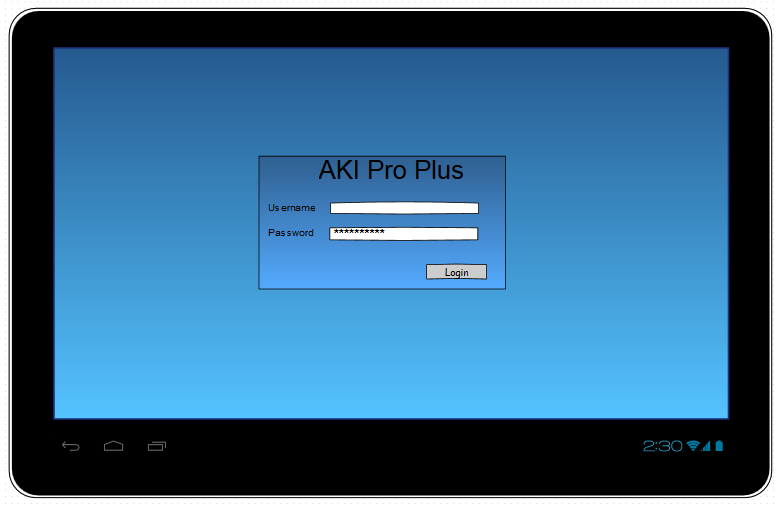
Driver Returns



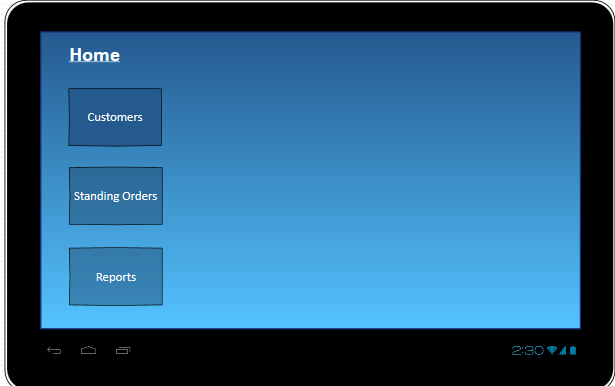
Driver Sync



**Sales Rep**

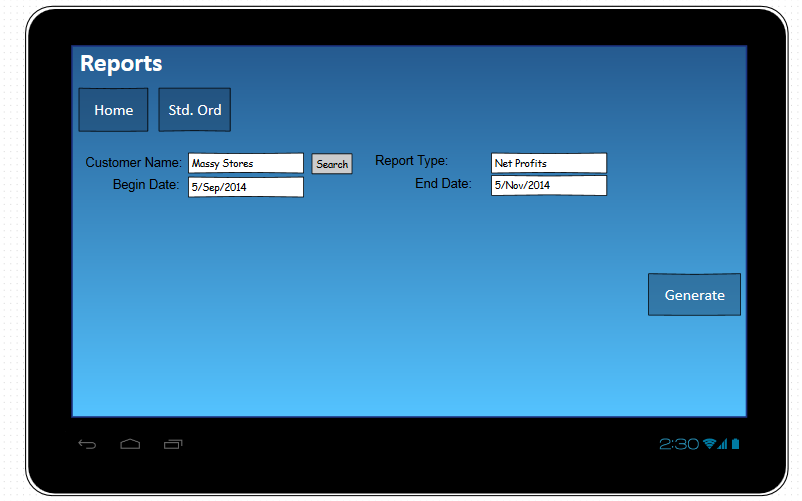
Sales Rep login

Sales Rep Home page



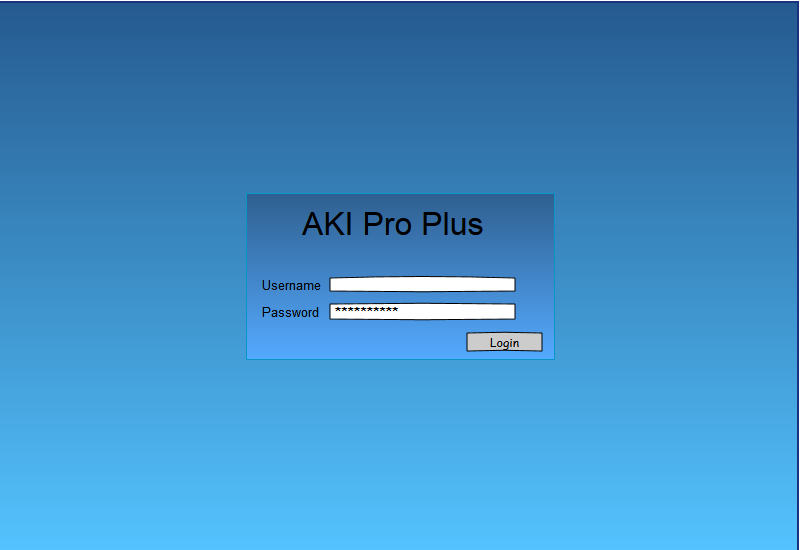
Sales Rep Standing Orders  


Sales Rep Reports

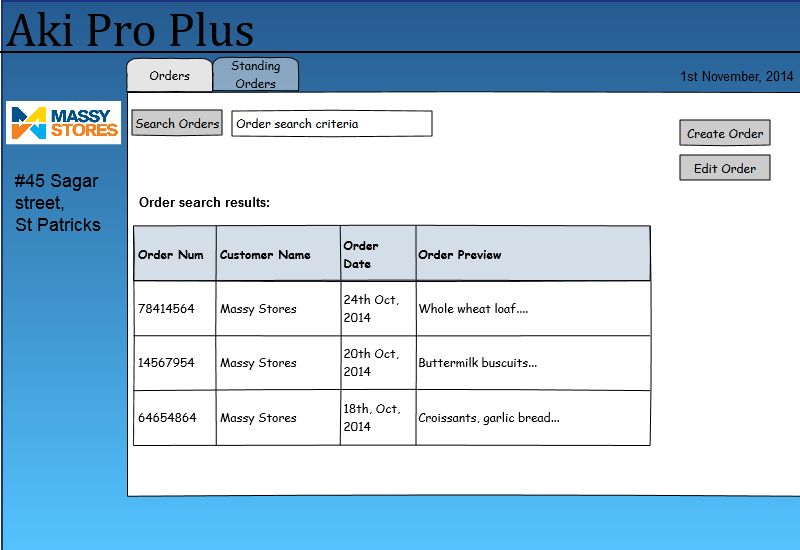


**Order Processing**

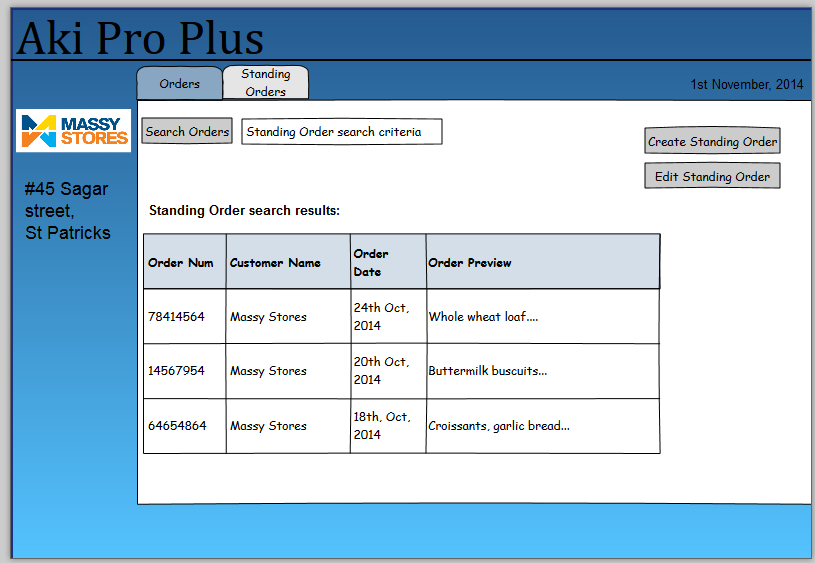
Login

****

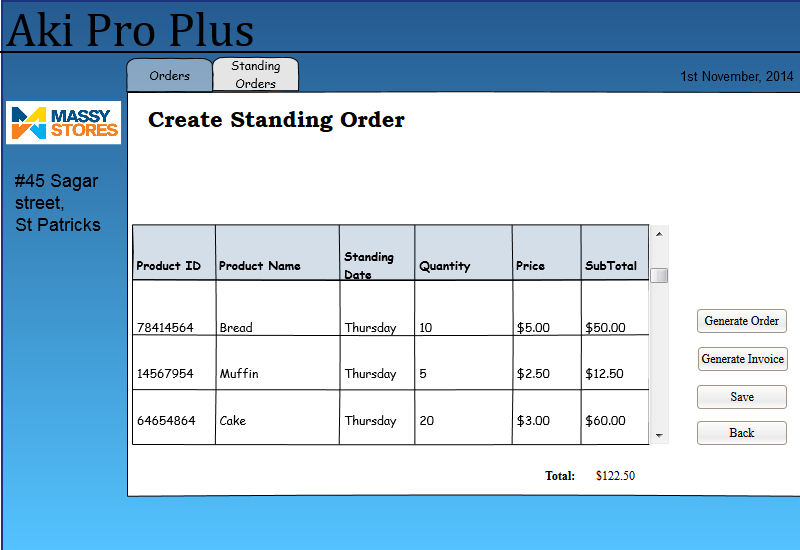
Default order screen after login

****

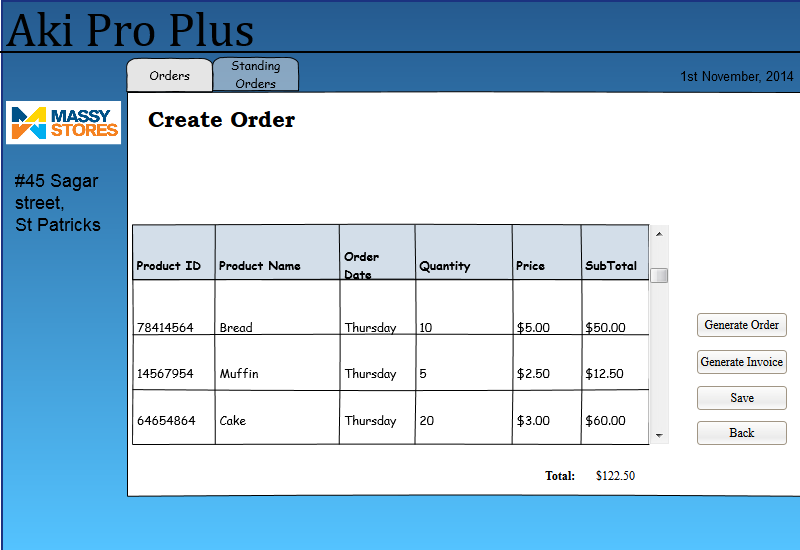
Standing Order tab



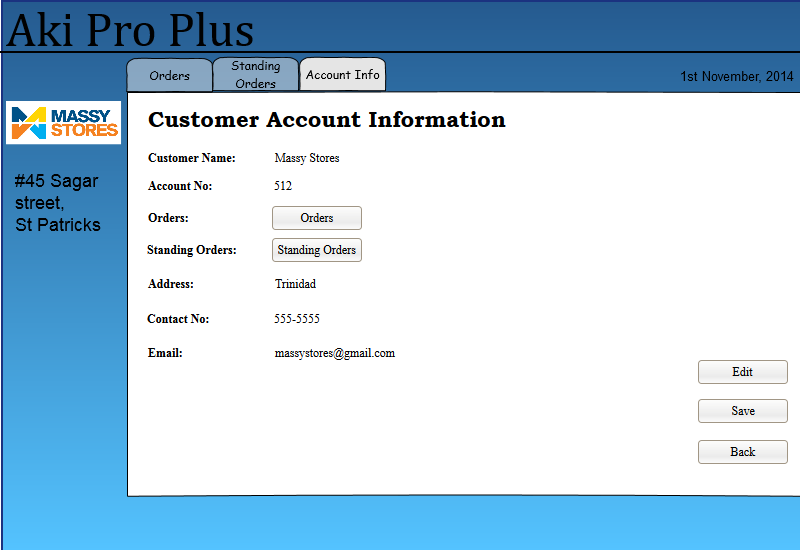
Create Standing Order option



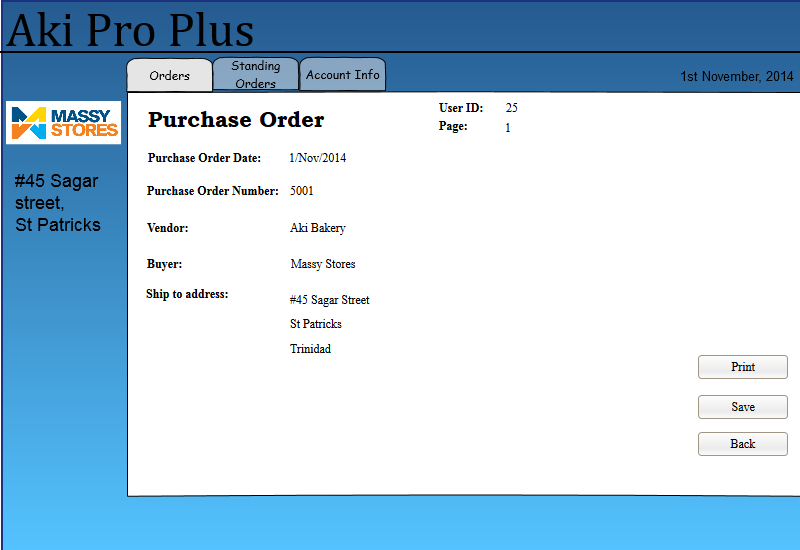
Create Order option



View Customer account info



View Purchase Orders



View Invoice Option

