Project team 4 - <Campus Courier>

Project Report

# Project Scope and Objectives

This project is intended to address a common problem on college campuses: many students find themselves without enough time to get food between classes, while others have extended periods of downtime during the day. The goal of this project is to provide the former group with a solution and the latter with a source of income by achieving the following:

1. Creating a centralized online menu for all on-campus a-la-carte dining.
2. Allowing students to place orders for food ahead of time.
3. Allowing other students to find work as “couriers”, who can earn money by picking up and delivering placed orders.
4. Creating a structure for these interactions that requires minimum direct supervision, similar to GrubHub or Uber.

# Project Environment

Our project uses ASP Web Forms, where the code-behind is written in C#. ASP Web Forms provide built-in universal browser compatibility, something we found important for a diverse set of student devices used to access the app. Our database access uses Entity Framework, which is a code-first SQL implementation that uses LINQ for queries and allows the developer to focus on writing business logic rather than data-access code. IIS (built-in with Visual Studio) was used as a web server.

We chose Visual Studio Community 2015 as our IDE, as Visual Studio is all-but-required for ASP.NET. Unfortunately, the choice to stay on the bleeding edge of VS updates broke our version control extension halfway through. Until that point, we used GitHub for version control, and we were able to fix the extension at the end of the project.

For testing, we used a combination of Visual Studio’s built-in unit testing framework and manual integration tests.

## SPRINT 0

**Story 30:**

**Description**:

## Prepare Development Environment

**Tasks**:

Task 31: [Download and Install Visual Studio Community 2015](https://tree.taiga.io/project/chaitub-team-4/task/31)

[Task 32:  Download and Install GitHub for Desktop, and link it to your GitHub account](https://tree.taiga.io/project/chaitub-team-4/task/32)

[Task 33: Download and Install MySQL Community Server](https://tree.taiga.io/project/chaitub-team-4/task/33)

[Task 34: Download and Install TeamCity](https://tree.taiga.io/project/chaitub-team-4/task/34)

[Task 35: Download and Install Sikuli](https://tree.taiga.io/project/chaitub-team-4/us/30?milestone=36262)

[Task 36: Create an account for InVision](https://tree.taiga.io/project/chaitub-team-4/task/36)

[Task 37: Create an account for Lumzy](https://tree.taiga.io/project/chaitub-team-4/us/30?milestone=36262)

[Task 38: Watch the video on the Moodle about using Taiga](https://tree.taiga.io/project/chaitub-team-4/task/38)

[Task 39: Complete the "First Points" requirements on Taiga](https://tree.taiga.io/project/chaitub-team-4/task/39)

**Story 40:**

**Description**:

**Create Initial User Stories**

Not sure if these can be included here//Create user stories that follow INVEST and describe project requirements. See attachment for (non-exhaustive) flowchart of general pipeline.

Keep these stories in the backlog, we will assign them to a sprint in an upcoming meeting.

We're using a modified fibonacci point system. Individual stories should be between 5-20 points (remember that we have 200 points for the whole project.

**SPRINT 1**

**Story 43:**

**Description**:

As a customer, I should be able to place an order for food from restaurants on campus

**Acceptance criteria**:

1. Customer selects the Food Store.

Given: Select List of Food Store: -- Select--

When: Customer Selects one Food Store from the List

Then: Food Store gets selected and Customer can see the list of food items.

2. Customer places order.

Given: List of Food items

When: Customer selects food items and press on "Place Your Order" button.

Then: Customer gets a confirmation message telling them your order is placed successfully and they can proceed for payment.

**Tasks**:

Task 71: Create "Order" Class

Task 72: Create "Orders" Database

Task 73: Implement "Check Out" functionality on Web App that inserts order into database

**Story 45**

**Description:**

As a customer, I should be able to view open dining locations and their menus

**Acceptance Criteria:**

1: Given: The user is on the site

When: The user clicks on the name of a restaurant

Then: The user may view the menu for that restaurant

2: Given: The user is viewing the menu

When: The user selects another restaurant from the top bar

Then: The user will view the other restaurant’s menu

**Tasks**:

Task 65: Create Framework for Web App using HTML5 and asp.net

Task 66: Create menu pages for 5 campus dining locations

**Story 55:**

**Description**:

As a courier, I should be able to view available deliveries

**Acceptance criteria**:

1: View List of deliveries

Given: Screen contains a "List of Deliveries" button

When: Courier press on the "List of Deliveries" button

Then: They can see the list of deliveries

2. Accept available deliveries

Given: "List of Deliveries" with "Deliver" Button

When: Courier presses on "Deliver" Button

Then: Courier gets a confirmation message, and that specific delivery is no more shown in the list.

**Tasks**:

[Task 74: Implement Database Access in Web App for Couriers to view "Orders"](https://tree.taiga.io/project/chaitub-team-4/task/74)

[Task 97: Make a GridView on the view and connect it with the database](https://tree.taiga.io/project/chaitub-team-4/task/97)

**Story 62:**

**Description**:

As a customer, I should be able to place orders by clicking options on a menu

**Acceptance criteria**:

1: Given: The user is viewing a menu

When: The user clicks on an item

Then: The item is added to the user’s cart

2: Given: The user is viewing their cart

When: The user selects “checkout”

Then: The user may pay and their order is placed

3: Given: The user is viewing their cart

When: The user selects “remove item”

Then: The item is removed from the user’s cart

**Tasks**:

Task [67: Create "Shopping Cart" class](https://tree.taiga.io/project/chaitub-team-4/task/67)

[Task 68: Implement interactivity on menu pages](https://tree.taiga.io/project/chaitub-team-4/task/68)

[Task 69: Implement shopping cart on site](https://tree.taiga.io/project/chaitub-team-4/task/69)

[Task 117: Create Unit Tests for Shopping Cart](https://tree.taiga.io/project/chaitub-team-4/task/117)

**Sprint 2**

**Story 79:**

**Description**:

As a customer, I want to be able to modify the contents of my shopping cart so that I may order items in quantity or remove items

**Acceptance criteria**:

Given: the customer is on the Shopping Cart page (shoppingCart.aspx)

When: the customer changes the quantity or remove field and clicks "Update"

Then: the customer's cart will be modified and displayed to them

**Tasks**:

Task 86: [Add "update" methods for quantity/removal fields of shopping cart class](https://tree.taiga.io/project/chaitub-team-4/task/86)

[Task 105: Create Unit Tests for this story](https://tree.taiga.io/project/chaitub-team-4/task/105)

**Story 50:**

**Description**:

As a dining location representative, I should be able to edit and update the posted menu

**Acceptance criteria**:

1: Given: Dining Representative edits the menu items

When: Clicks on "Update" button.

Then: Dining Representative can see a confirmation message saying "Changes in Menu are saved Successfully".

2: Given: Dining Representative can see menu items and "Edit" button

When: Dining Representative clicks on the button

Then: Menu descriptions gets open to be edited.

**Tasks**:

Task 100: Create "representative" view for Product Database

Task 101: Create update methods for restaurant menus

Task 106: Create Unit Tests for this story

**Story 48:**

**Description**:

As a courier, I should be able to view orders to which I have been assigned

**[Tasks](https://tree.taiga.io/project/chaitub-team-4/task/91" \o "#91 Create \"active order\" view for Courier users (1 hour))**[:](https://tree.taiga.io/project/chaitub-team-4/task/91" \o "#91 Create \"active order\" view for Courier users (1 hour))

[Task 91: Create "active order" view for Courier users](https://tree.taiga.io/project/chaitub-team-4/task/91" \o "#91 Create \"active order\" view for Courier users (1 hour))

**Story 53**

**Description**:

As a courier, I should be able to provide confirmation that the customer has received their order

**Acceptance criteria**:

Given: the courier is viewing their pending orders

When: the courier clicks the "delivered" button next to an order

Then: the customer is prompted to confirm the delivery

**Tasks:**

Task [94: Create confirmation page that contains order info](https://tree.taiga.io/project/chaitub-team-4/task/94)

[Task 107: Create Unit Tests for this story](https://tree.taiga.io/project/chaitub-team-4/task/107)

**Story 57**

**Description**:

As a customer, I should be able to confirm that my order was delivered

**Acceptance criteria**:

Given: the courier is viewing a list of available orders

When: the courier selects "deliver this order"

Then: the order is added to the courier's pending list of deliveries

**Tasks:**

Task [88: Implement Shopping Cart class for couriers to add orders to a "cart" to be accepted](https://tree.taiga.io/project/chaitub-team-4/task/104" \o "#104 Add \"confirm delivery\" to customer post-checkout view (1-2 hours))

[Task 108: Create Unit Tests for this story](https://tree.taiga.io/project/chaitub-team-4/task/112)

**Story 58**

**Description**:

As a customer, I should be able to add a tip to my payment

**Acceptance criteria**:

Given: the customer is checking out

When: the customer is choosing their payment method and adds a tip to the tip field

Then: the tip is added to the customer's total before payment is processed

**Tasks**:

[Task 93: Implement "tip" field when customer total is calculated](https://tree.taiga.io/project/chaitub-team-4/task/93)

[Task 98: Create a payment page, can also add paypal extension](https://tree.taiga.io/project/chaitub-team-4/task/98)

[Task 110: Create Unit Tests for this story](https://tree.taiga.io/project/chaitub-team-4/task/110)

**Story 46**

**Description**:

As a customer, I should be able to receive confirmation that my order has been placed

**Acceptance criteria**:

Given: Customer places order to a Food Store

When: Select List of Food Store: -- Select--

Select item from the menu: --Select

Make payment: --pay--

Confirm the order: Would you like to confirm the order?

Then: The customer receives a message to the registered phone number that the order has been placed successfully

**Tasks:**

Task 70: Implement confirmation message that appears when an order is placed

**Story 83**

**Description**:

As a courier, I should be able to accept deliveries to which I want to be assigned

**Acceptance criteria**:

Given: Customer can see a list of orders and "Received Order" button.

When: Customer clicks "Received Order" button

Then: Customer can see a confirmation message - "Thanking for using Campus Courier... See you Soon..!!!!!"

Given: Option "Your Orders" to view all the orders

When: Customer selects "Your Orders"

Then: Customer can see list of their orders starting from the recent one.

**Tasks**

Task [104: Add "confirm delivery" to customer post-checkout view](https://tree.taiga.io/project/chaitub-team-4/task/104)

[Task 112: Create Unit Tests for this story](https://tree.taiga.io/project/chaitub-team-4/task/112)

**SPRINT 3**

**Story 49**

**Description**:

As a dining location representative, I should be able to view orders sent to my location

**Acceptance criteria**:

Given: Customers make orders to a Food courts, Dining representative would like to view orders

When: Dinning location representatives will be able to view orders made to it:--View Orders--

Then: Food court can keep track of orders made to its location

**Tasks**:

Task 77: Create view panel for restaurant locations to view orders

**Story 47**

**Description**: As a courier, I should be able to create an account to track my pending and completed deliveries

**Acceptance criteria**:

Given: the courier is on the site

When: the courier clicks "view orders"

Then: the courier is shown a list of available orders to accept

Given: the courier is on the site

When: the courier clicks "completed orders"

Then: the courier is shown a list of their completed orders

**Tasks:**

Task 89: Implement Courier user role

Task 90: Create view for Orders database entries associated with courier's user ID

**Story 56**

Description

As a courier, I should be able to confirm that an order has been picked up

**Acceptance Criteria:**

Given- The courier picks up the delivery

When- The courier wants to confirm his pick up,

he clicks on the delivery pick up option --pick up--

Then- Status of Delivery changes from 'Reserved' to 'In progress'

The other couriers will not be able to accept deliveries that are already

reserved, in-progress or waiting for delivery

**Tasks:**

**Task 118**: Add a button "Pick up" on the Orderspage.aspx and change value of column "Status" in the Orders table

**Story 121**

**Description**: Implement Registration and Login

**Tasks**:

Task:123 Create user roles and implement asp's built-in login

**Story 115**

**Description**: Menu items need images

**Tasks:**

Task 116: Add images for menu items

## Story 130

## Description: Integrate solutions for final demo

## Story 134

## Description: Hide role-specific pages from general view

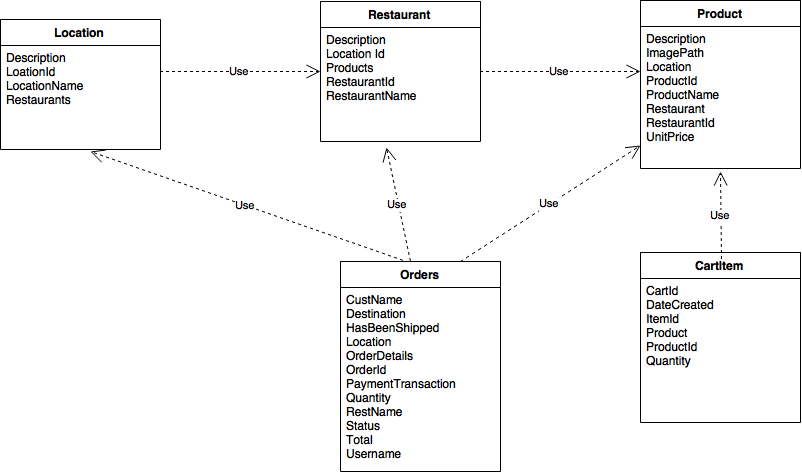
## Software Design

### **HIGH LEVEL ARCHITECTURE**

We have used Entity Framework Architecture for our application. The Entity Framework is a set of technologies in ADO.NET that support the development of data-oriented software applications. The framework supports having different layers for our business logic, Physical logic and presentation layer. We had our Physical logic in a folder named Logic which dealt with all the database connectivity stuff and we just created objects of the classes and used them. The advantage of using this architecture is it handles all the SQL stuff and the programmer does not have to worry about the tables and other database related things. Entity framework helps user to work at high level of abstraction with respect to data and hence for such this application we ended up writing very less code.

Entity Framework enables easy mapping object to data. For example for the Order Class, there were other classes using it like Orderdetails, Customer however this framework enables us Flexibility to represent this relationship and use it.

### **DETAILED CLASS DIAGRAM**



## Software Testing

As discussed in our presentation, ASP’s Web Forms are not built for unit tests. Due to the high level of interdependency inherent to the architecture, unit testing is almost-impossible except at a very basic level. Most of our testing was done manually during the integration phases, and then again during the project’s finalization. White box testing was used extensively, with some black box testing during the final Sprint.

### **SAMPLE TESTS**

**Test 1: Unit Test**

Requirement(s) under test: Story #62

Function(s)/Module(s) under test: ShoppingCartActions (Shopping Cart)

Initial conditions: there exist any number of carts in the database

Assumptions: the current user has a unique http session

Test case input: new ShoppingCartActions

Expected result: cartID is unique, itemCount = = 0

Result: Passed

**Test 2: Unit Test**

Requirement(s) under test: Story #50

Function(s)/Module(s) under test: AddProducts (Menu Editing)

Initial conditions: local DB is instantiated

Assumptions: valid input

Test case input: AddProduct(n, d, p, r, i), where n, d, p, r, and i are strings

Expected result: Product is inserted into database with above values (parsed when necessary)

Result: Passed

**Test 3: White Box Integration Test**

Requirement(s) under test: Story #50

Function(s)/Module(s) under test: AddProducts

Initial conditions: local DB is initialized, app is running

Assumptions: none in particular

Test case input: Fill in fields on EditMenu page, submit

Expected result: product is added to database

Result: Passed

**Test 4: Black Box Manual Test**

Requirement(s) under test: Story #62

Function(s)/Module(s) under test: ShoppingCart, Checkout, PayPal API, Login

Initial conditions: user is logged in

Assumptions: none

Test case input: add products to cart, Checkout,

Expected result: Order confirmation page, order added to DB

Result: Passed