ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτοματοποιημένο Σύστημα Ελέγχων για την βελτίωση της ποιότητας Ευέλικτων Έργων Λογισμικού

Του φοιτητή

Μποζαμπαλίδη Ιορδάνη

Αρ. Μητρώου: 08/3331

Επιβλέπων καθηγητής

Παναγιώτης Σφέτσος

Θεσσαλονίκη 2013

ΠΡΟΛΟΓΟΣ

Ένα μεγάλο σύστημα λογισμικού αναπτύσσεται γρήγορα και αλλάζει μορφή συχνά για να καλύψει τις ανάγκες του αγοραστικού του κοινού. Ένα τέτοιο σύστημα πολλές φορές αναπτύσσεται από ομάδες προγραμματιστών. Επομένως ο σωστός διαμοιρασμός αρμοδιοτήτων και υποχρεώσεων σε μία ομάδα ανάπτυξης είναι πολύ σημαντικό κομμάτι της σχεδίασης. Μία τέτοια ομάδα αναπτύσσει το σύστημα POS (Point of Sales) της εταιρίας Foodtec Solutions.

Η συντήρηση και εξέλιξη όλου αυτού του έργου είναι ζωτικής σημασίας μιάς και ακολουθείται η τεχνική της ευέλικτης σχεδίασης. Κάθε αλλαγή που προστίθεται στο σύστημα πρέπει να ελεγχθεί και να διασφαλιστεί η σωστή λειτουργία του συνολικού συστήματος.

Αυτή η διαδικασία μπορεί να γίνει εύκολα με την χρήση αυτοματοποιημένων συστημάτων ελέγχου, χρησιμοποιώντας τις νέες τεχνολογίες.

Η εργασία θα καλύψει την συγκεκριμένη διαδικασία και θα παρουσιάσει αυτά τα συστήματα.

ΠΕΡΙΛΗΨΗ

Αρχικά θα παρουσιαστεί το συστήμα για το οποίο θα υλοποιήσουμε τα case studies. Μία αναλυτική περιγραφή του συστήματος με τις λειτουργίες του και τις δυνατότητές του.

Στη συνέχεια θα εμβαθύνουμε στην αρχιτεκτονική και δυνατότητες του συστήματος πάνω στο οποίο θα υλοποιηθεί η αυτοματοποίηση. Θα γίνει παρουσίαση των τεχνικών μερών του και κατηγοριοποίηση των αντικειμένων πάνω στα οποία θα εφαρμοστούν οι διαδικασίες αυτοματοποιημένου ελέγχου.

Θα ακολουθήσει η παρουσίαση των εργαλείων και των δυνατοτήτων τους. Για την αυτοματοποίηση θα χρησιμοποιήσουμε το Selenium το οποίο θα μας δώσει τη δυνατότητα να χειριστούμε τα αντικείμενα (web elements) έτσι ώστε να αυτοματοποιήσουμε τις διαδικασίες ελέγχου. Η σύνταξη των tests θα γίνει με την βοήθεια του γνωστού εργαλείου JUnit.

Στη συνέχεια θα γίνει περιγραφή των βιβλιοθηκών που προσφέρονται από το Selenium και των έτοιμων μεθόδων που δίνουν την δυνατότητα στην εύκολη και γρήγορη διαχείριση των στοιχείων που μας ενδιαφέρουν.

Θα υπάρξει στην συνέχεια η ανάπτυξη των Test Cases. Θα παρουσιαστούν αναλυτικά όλα τα Test Cases για το Web Ordering του συστήματος.

Θα παρουσιαστούν τα σημαντικότερα κομμάτια από τον κώδικα Java ο οποίος αποτελεί το project βασισμένο πάνω στο Selenium. Αυτά τα κομμάτια θα παρουσιάσουν τις σημαντικότερες λειτουργίες και θα επεξηγούνται αναλυτικά.

Τέλος θα εμφανιστούν συνοπτικά τα αποτελέσματα και η βελτίωση της απόδοσης του χρόνου.

ABSTRACT

Automated testing is fundamental for projects that are being developed with agile methods. In a big fast developed project, automated testing can be crucial for the final result and quality of the product.

The project that is described below is a web platform for ordering. This product has many aspects and functionality. In order to develop such a product the company needs to use practices that will ensure the quality of the product.

The tools that are going to be used are the Selenium Web Driver and the JUnit framework. These tools give us the ability to handle and use web components in order to perform a series of tests.

We will present the classes and method that can handle the web components of the web pages and can help us to compose the tests. Our goal is to have the required infrastructure in order to be able to create and run easily and fast any tests.

The methods used are focused on Acceptance testing. This methods ensures the quality of the final product and guarantee that all the customers requirements are met.

Finally we compare the manual and the automated procedure and present the benefits of each method.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά τον καθηγητή μου κ Παναγιώτη Σφέτσο για την υποστήριξή του για αυτή την εργασία.

Ευχαριστώ τα μέλη της Foodtec που μου έδωσαν την δυνατότητα να υλοποιήσω την πτυχιακή μου εργασία πάνω στο σύστημα τους.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Η ομάδα ανάπτυξης χωρίζεται σε δύο μέρη. Τους προγραμματιστές (developers) και τους testers. Κάθε νέα προσθήκη ή αλλαγή-βελτίωση του κώδικα πρέπει να ελεγθεί ως προς την χρήση και την λειτουργία του. Για την ανάπτυξη του συστήματος ακολουθείται η ευέλικτη σχεδίαση (agile development). Κάθε νέα προσθήκη ενσωματώνεται στο υπάρχον σύστημα και ελέγχεται από την ομάδα του QA (Quality Assurance), ούτως ώστε να εξασφαλιστεί η ποιότητα του τελικού προιόντος.

Για μία τέτοια προσέγγιση στο τρόπο σχεδιασμού χρειάζεται ξεκάθαρη γνώση του συστήματος της λειτουργίας του συστήματος. Είναι απαραίτητο να υπάρχουν ξεκάθαρες αρχές οι οποίες ακολουθούνται έτσι ώστε το προϊόν να μην αποκλίνει από την σωστή λειτουργία. Για αυτό είναι σημαντικό να ορίζονται περιπτώσεις χρήσης για κάθε πεδίο του συστήματος. Με αυτόν τον τρόπο, έχοντας μια σειρά από βήματα που πρέπει να ακολουθηθούν, ο κάθε tester μπορεί να κρίνει αν μία νέα αλλαγή στον κώδικα είναι σωστή και μπορεί να προστεθεί στη νέα έκδοση του συστήματος.

Η διαδικασία εξέλιξης του λογισμικού που ακολουθείται είναι η εξής. Κατ αρχήν περιγράφεται σε ένα νέο bug μία αλλαγή που έχει ζητήσει κάποιος πελάτης, κάποιο πρόβλημα στη χρήση του λογισμικού που έχει εντοπίσει ένα χρήστης ή μία νέα προσθήκη βελτίωσης του συστήματος enhancement. Με αυτόν τον τρόπο είναι ξεκάθαρο τί πρέπει να γίνει και με ποιόν τρόπο αυτή η αλλαγή θα επιρεάσει τον τελικό χρήστη. Στη συνέχεια αυτό το bug θα ανατεθεί σε έναν developer για να αξιολογήσει το πρόβλημα και να αποφασίσει με ποιόν τρόπο θα το επιλύσει και ποιούς τομείς του λογισμικού θα επηρεάσει Αφού προσθέσει την αλλαγή ο developer (submit), ο κώδικας με τις αλλαγές του θα περάσει από αξιολόγηση (review) ούτως ώστε να επιβεβαιωθεί ότι οι αλλαγές είναι χρήσιμες και η χρήση των εντολών είναι η κατάλληλη. Αφού ολοκληρωθεί και αυτή η φάση το bug είναι έτοιμο για έλεγχο. Ο tester πλέον πρέπει να ελέγξει την λειτουργικότητα της νέας αλλαγής και να κρίνει αν χρειάζεται βελτίωση, αν έχει επηρεάσει αρνητικά κάποια άλλη λειτουργία του υπολοίπου συστήματος και αν ανταποκρίνεται το τελικό αποτέλεσμα στην αρχική εκτίμηση της αλλαγής. Αν όλα είναι σωστά τότε το bug μαρκάρεται σαν tested και είναι έτοιμο να προστεθεί στην επόμενη έκδοση του συστήματος.

Μία έκδοση αποτελείται από πολλά bugs. Υπάρχουν εκδόσεις με απλές διορθώσεις και εκδόσεις με σημαντικές βελτιώσεις και προσθήκες (major releases). Όταν μία έκδοση χαρακτηρίζεται ως major είναι σημαντικό να ελεγχθεί ολόκληρο το σύστημα ως προς την ορθή λειτουργία του. Το σύστημα χωρίζεται σε κομμάτια (modules) τα οποία ενωμένα μαζί αποτελούν το τελικό προϊόν Επομένως όταν μία έκδοση είναι major πρέπει να γίνει ένα regression test σε όλα τα modules. Σε αυτή τη διαδικασία πρέπει να τρέξουν όλα τα test cases για όλα τα modules για να εξασφαλιστεί ότι το τελικό προϊόν θα λειτουργεί και θα συμπεριφέρεται έτσι όπως θα έπρεπε.

Όλη αυτή η διαδικασία είναι εξαιρετικά χρονοβώρα και είναι πολύ σημαντικό για την εταιρία να γίνεται αποδοτικά. Η αυτοματοποίηση όλης αυτής της διαδικασίας είναι ζωτικής σημασίας για ένα σύστημα που εξελίσεται ραγδαία και συνεχώς ενσωματώνονται καινούργιες αλλαγές. Σκοπός αυτής της εργασίας είναι να αυτοματοποιηθεί το QA regression testing του Web Ordering module του συστήματος. Με αυτόν τον τρόπο ένα ζωτικό κομμάτι του συστηματος που είναι το online web ordering μπορεί να έχει ένα αυτόματο έλεγχο της λειτουργίας του πολύ απλά με τον πάτημα ενός κουμπιού.

Υπάρχουν πολλά εργαλεία που δίνουν την δυνατότητα στον tester να δημιουργήσει ένα τέτοιο αυτοματοποιημένο περιβάλλον. Στην συγκεκριμένη εργασία θα επικεντρωθούμε στην χρήση Selenium το οποίο μας δίνει την δυνατότητα να χειριστούμε έναν browser με εντολές Java.

Στα παρακάτω μέρη θα παρουσιαστούν αναλυτικά τα εργαλεία και οι δυνατότητές τους, θα παρουσιαστεί το σύστημα ως προς την χρήση του, τα test cases για το module και η δημιουργία του αυτοματοποιημένου συστήματος.

ΚΕΦΑΛΑΙΟ 1

Παρουσίαση του συστήματος.

ΕΙΣΑΓΩΓΗ

Το κομμάτι που θα καλύψουμε και θα αναλύσουμε είναι το Web Ordering της εταιρίας. Αυτό αποτελεί το κομμάτι που αλληλεπιδρά ο χρήστης και μπορεί να πραγματοποιήσει τις αγορές του μέσω ίντερνετ. Αυτό αποτελείται από δύο μέρη, τον server και το web ordering από το οποίο μπορεί να πραγματοποιήσει τις αγορές του ο πελάτης.

Το σύστημα αυτό απευθύνεται κυρίως σε εστιατόρια και πιο συγκεκριμένα σε πιτσαρίες. Το συγκεκριμένο σύστημα που θα παρουσιαστεί είναι από μία μεγάλη αλυσίδα εστιατορίων των Η.Π.Α. Το σύστημα αυτό δίνει την δυνατότητα για παραγγελία μέσω διαδικτύου εύκολα και γρήγορα και την άμεση ειδοποίηση του καταστήματος για τις παραγγελίες που έχουν πραγματοποιηθεί.

Παρακάτω θα παρουσιαστεί αναλυτικά κάθε μέρος του συστήματος. Κάθε σελίδα στην οποία μπορεί να αλληλεπιδράσει ο χρήστης έχει διαφορετικές λειτουργίες. Όλες αυτές οι λειτουργίες θα παρουσιαστούν αναλυτικά παρακάτω.

ΥΠΟΚΕΦΑΛΑΙΟ 1.1

Intro Page

Αυτή η σελίδα είναι η πρώτη που θα αντικρίσει ο χρήστης μόλις εισέλθει στο σύστημα. Μπορεί να επιλέξει το είδος της παραγγελίας πατώντας στα radio buttons. Υπάρχουν δύο διαθέσιμα είδη παραγγελίας Delivery και Carry Out. Κάθε επιλογή εμφανίζει ένα νέο μενού.

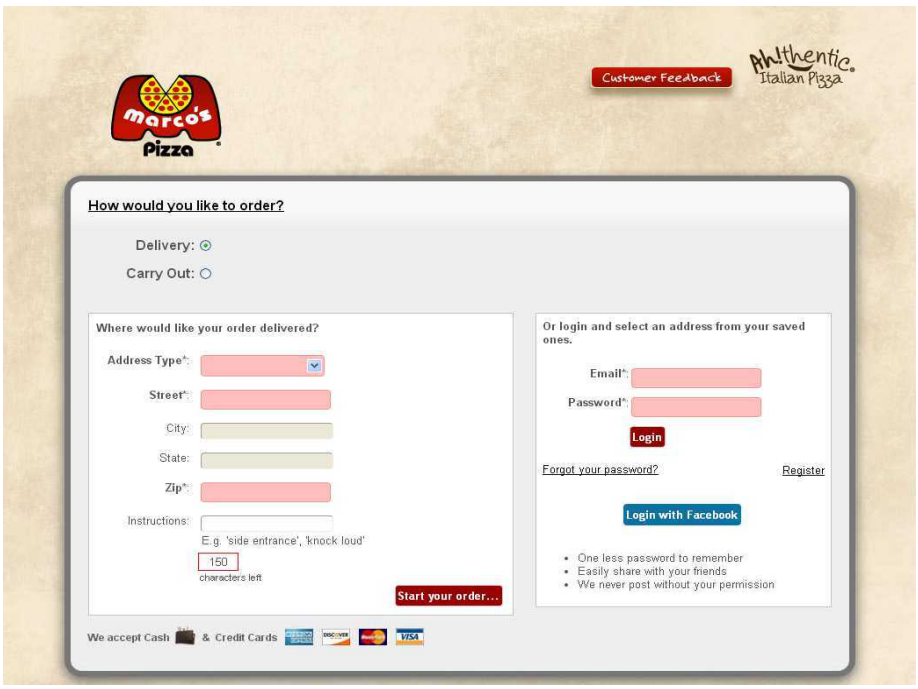
1.Intro Page

Εάν επιλεχθεί το Carry Out, στον χρήστη θα εμφανιστεί ένα κουμπί που του επιτρέπει να κατευθυνθεί στην επόμενη σελίδα και να ξεκινήσει την παραγγελία του.

2.Intro Carry Out

Αν ο χρήστης επιλέξει Delivery εμφανίζεται ένα νέο dialog απ'όπου ο χρήστης μπορεί να εισάγει την διεύθυνση του έτσι ώστε να χρησιμοποιηθεί για να ολοκληρώσει την παραγγελία του. Υπάρχει η δυνατότητα στο πεδίο της διεύθυνσης ο χρήστης εισάγοντας γράμματα από την διεύθυνσή του να εμφανίζονται προτάσεις και υποδείξεις γι αυτό που θέλει να εισάγει(auto-complete).

Το δεύτερο dialog που εμφανίζεται δίνει την δυνατότητα στον χρήστη να κάνει log in στο account του είτε με χρήση του Facebook είτε με τα credentials του χρήστη. Μπορεί επίσης να εγγραφεί ή να επιλέξει υπενθύμιση του κωδικού του.



3.Intro Delivery

ΥΠΟΚΕΦΑΛΑΙΟ 1.2

Main Menu Page

Από αυτή τη σελίδα ο χρήστης μπορεί να ξεκινήσει την παραγγελία του επιλέγοντας αντικείμενα και παραμετροποιώντας την παραγγελία του με πολλούς διαθέσιμους τρόπους.

Υπάρχει διαθέσιμο ένα κεντρικό μενού (Navigation Menu) απ'όπου ο χρήστης έχει την δυνατότητα να κατευθυνθεί στις ακόλουθες σελίδες:

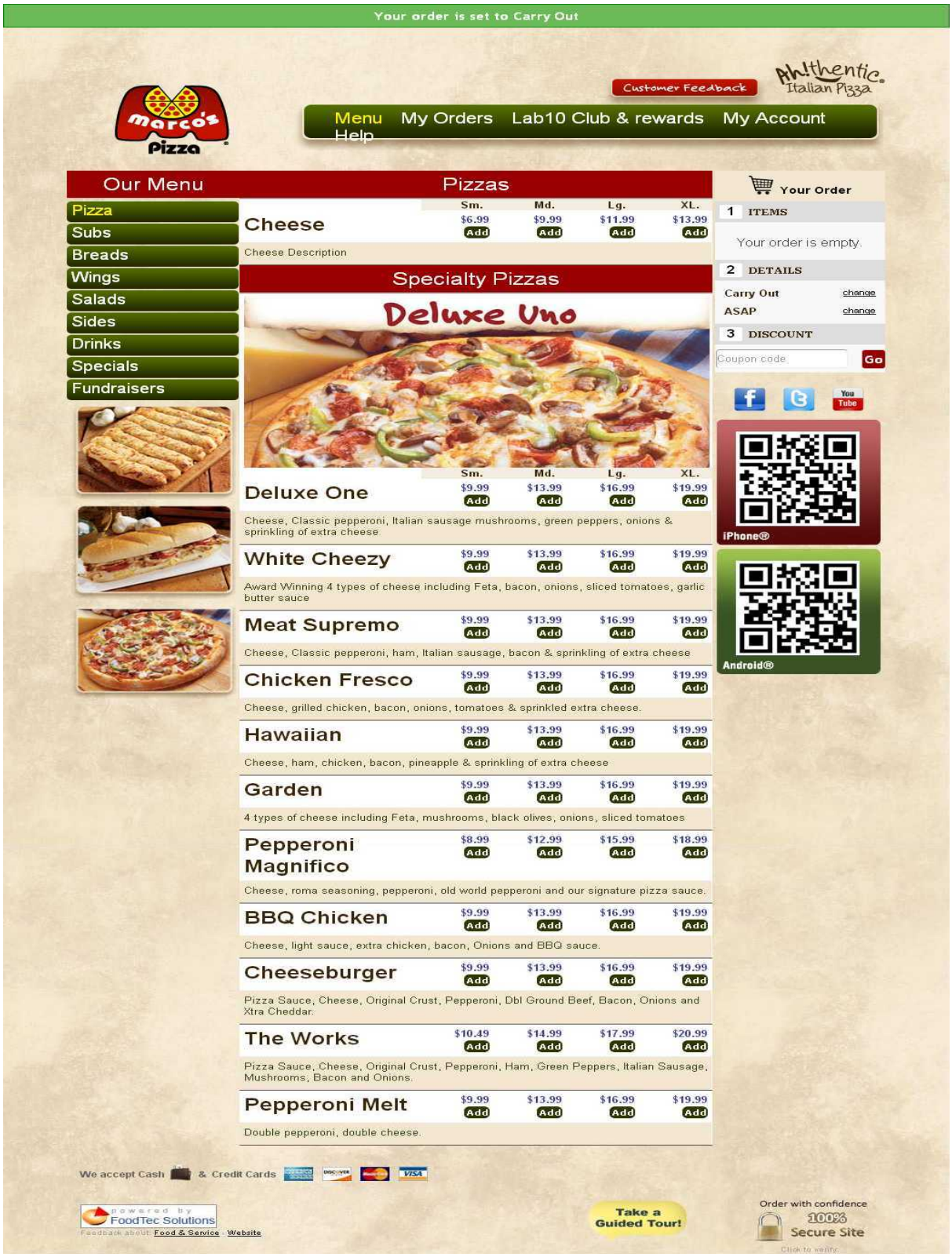
- Menu
- My Orders
- Loyalty Club and Rewards
- My Account
- Help

Η Menu Page αποτελείται από τρία columns.

Το αριστερό column περιέχει τις κατηγορίες του μενού και εμφανίζει μία λίστα με όλες τις διαθέσιμες κατηγορίες που μπορεί να επιλέξει ο χρήστης.

Το κεντρικό column περιέχει τα αντικείμενο για κάθε επιλεγμένη κατηγορία καθώς επίσης δίνει και την επιλογή για παραμετροποίηση του επιλεγμένου αντικειμένου. Εμφανίζει τα διαθέσιμα μεγέθη για κάθε αντικείμενο και την τιμή του κάθε αντικειμένου. Κάθε κατηγορία έχει διαφορετικά αντικείμενα και επιλογές.

Το δεξιό column εμφανίζει το καλάθι αγορών. Το συγκεκριμένο column είναι μια περίληψη της παραγγελίας. Υπάρχει μία λίστα με τα αντικείμενα που έχει προσθέσει ο χρήστης στο καλάθι αγορών του, το συνολικό ποσό, καθώς επίσης και πολλές επιλογές για την παραγγελία όπως αν επιθυμεί να χρήστης να είναι Delivery/Carry Out, αν θέλει ο χρήστης να χρησιμοποιήσει κάποιο κουπόνι, αν θέλει να την παραλάβει άμεσα η μία προκαθορισμένη ώρα και η δυνατότητα να ακυρώσει την παραγγελία του.

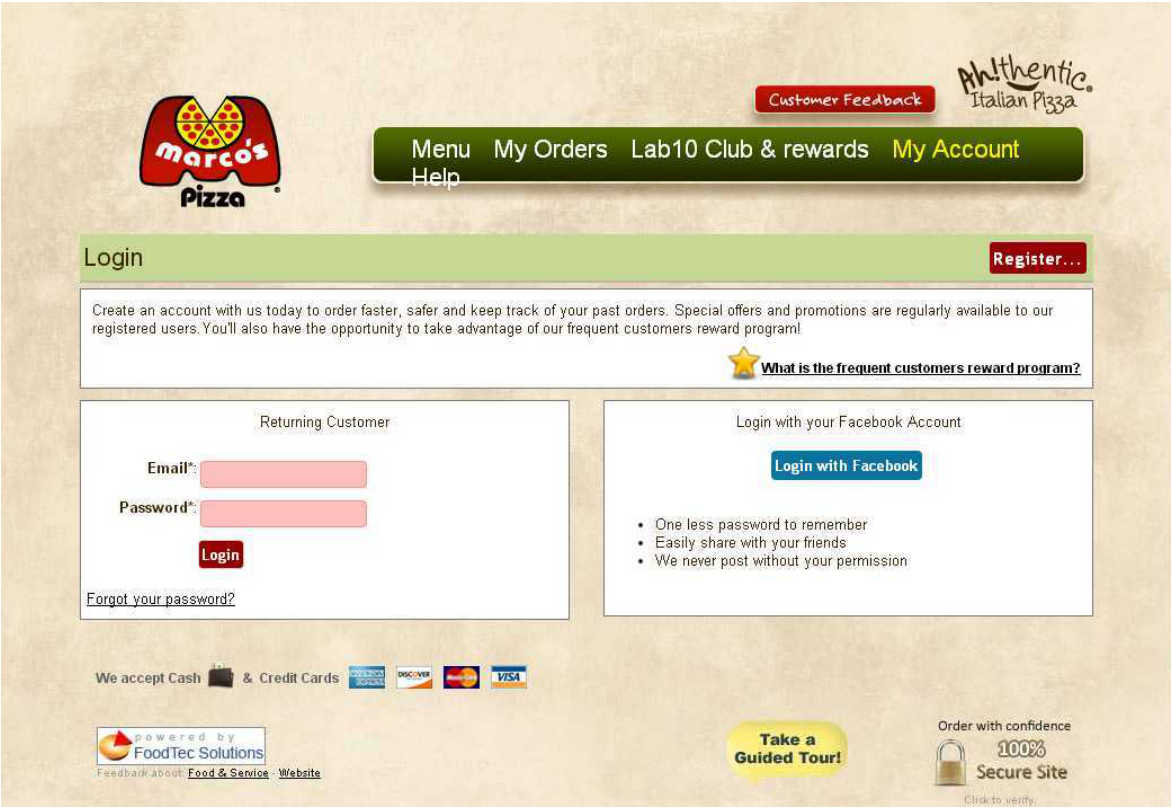

Your order is set to Carry Out
Customer Feedback
Authentic Italian Pizza
Menu My Orders Lab10 Club & rewards My Account Help
Our Menu
Pizza
Subs
Breads
Wings
Salads
Sides
Drinks
Specials
Fundraisers
Pizzas
Sm. Md. Lg. XL.
Cheese $6.99 $9.99 $11.99 $13.99 Add
Cheese Description
Specialty Pizzas
Deluxe Uno
Deluxe One $9.99 $13.99 $16.99 $19.99 Add
Cheese, Classic pepperoni, Italian sausage mushrooms, green peppers, onions & sprinkling of extra cheese
White Cheezy $9.99 $13.99 $16.99 $19.99 Add
Award Winning 4 types of cheese including Feta, bacon, onions, sliced tomatoes, garlic butter sauce
Meat Supremo $9.99 $13.99 $16.99 $19.99 Add
Cheese, Classic pepperoni, ham, Italian sausage, bacon & sprinkling of extra cheese
Chicken Fresco $9.99 $13.99 $16.99 $19.99 Add
Cheese, grilled chicken, bacon, onions, tomatoes & sprinkled extra cheese.
Hawaiian $9.99 $13.99 $16.99 $19.99 Add
Cheese, ham, chicken, bacon, pineapple & sprinkling of extra cheese
Garden $9.99 $13.99 $16.99 $19.99 Add
4 types of cheese including Feta, mushrooms, black olives, onions, sliced tomatoes
Pepperoni Magnifico $8.99 $12.99 $15.99 $18.99 Add
Cheese, roma seasoning, pepperoni, old world pepperoni and our signature pizza sauce.
BBQ Chicken $9.99 $13.99 $16.99 $19.99 Add
Cheese, light sauce, extra chicken, bacon, Onions and BBQ sauce.
Cheeseburger $9.99 $13.99 $16.99 $19.99 Add
Pizza Sauce, Cheese, Original Crust, Pepperoni, Dbl Ground Beef, Bacon, Onions and Xtra Cheddar.
The Works $10.49 $14.99 $17.99 $20.99 Add
Pizza Sauce, Cheese, Original Crust, Pepperoni, Ham, Green Peppers, Italian Sausage, Mushrooms, Bacon and Onions.
Pepperoni Melt $9.99 $13.99 $16.99 $19.99 Add
Double pepperoni, double cheese.
Your Order
1 ITEMS
Your order is empty.
2 DETAILS
Carry Out change
ASAP change
3 DISCOUNT
Coupon code
Go
iPhone®
Android®
We accept Cash & Credit Cards
powered by FoodTec Solutions
Take a Guided Tour!
Order with confidence
100% Secure Site

4.Menu Page

ΥΠΟΚΕΦΑΛΑΙΟ 1.3

Login Page

Η συγκεκριμένη σελίδα δίνει στον χρήστη την δυνατότητα να συνδεθεί με τον λογαριασμό του εύκολα και γρήγορα, είτε να εγγραφεί στο σύστημα για να μπορεί να χρησιμοποιεί της δυνατότητες που του προσφέρει ο λογαριασμός χρήστη. Επίσης μπορεί να χρησιμοποιήσει τον λογαριασμό του Facebook έτσι ώστε να είναι ευκολότερη και πιό γρήγορη η πρόσβαση.



5.Login Page

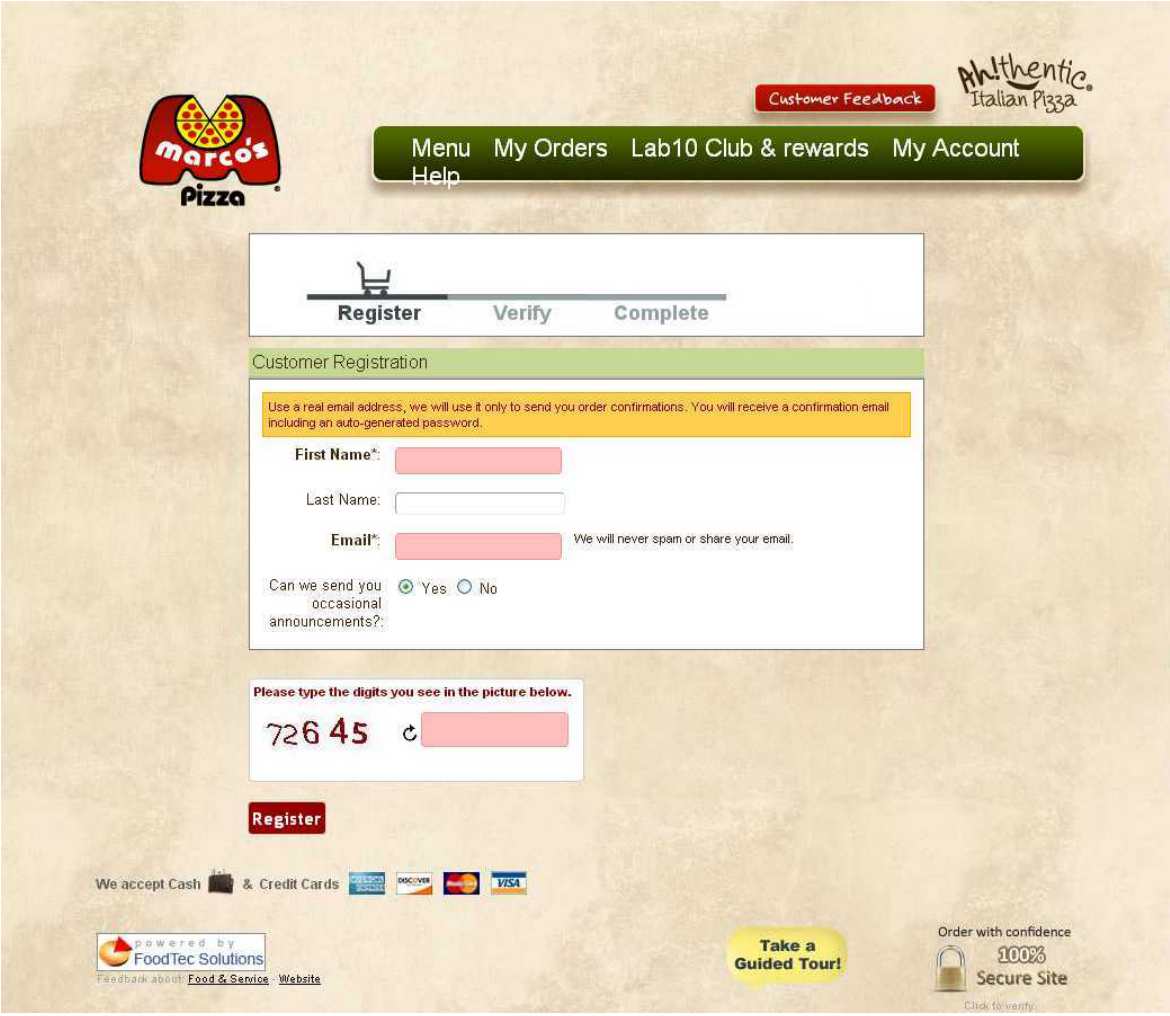
ΥΠΟΚΕΦΑΛΑΙΟ 1.4

Register Page

Ο χρήστης μπορεί να δημιουργήσει έναν λογαριασμό χρήστη έτσι ώστε να μπορεί να αποθηκεύει χρήσιμες πληροφορίες και να μπορεί επίσης να κερδίζει προσφορές. Από την register page ο χρήστης μπορεί να γραφτεί εύκολα και γρήγορα.

Ο χρήστης θα χρειαστεί να εισάγει το ονοματεπώνυμό του και το email του, να επιλέξει αν θέλει να γραφτεί στο newsletter και να εισάγει έναν κωδικό captcha.

Αμέσως μόλις εισάγει τις απαραίτητες πληροφορίες θα λάβει ένα email για να επιβεβαιώσει την εγγραφή του.



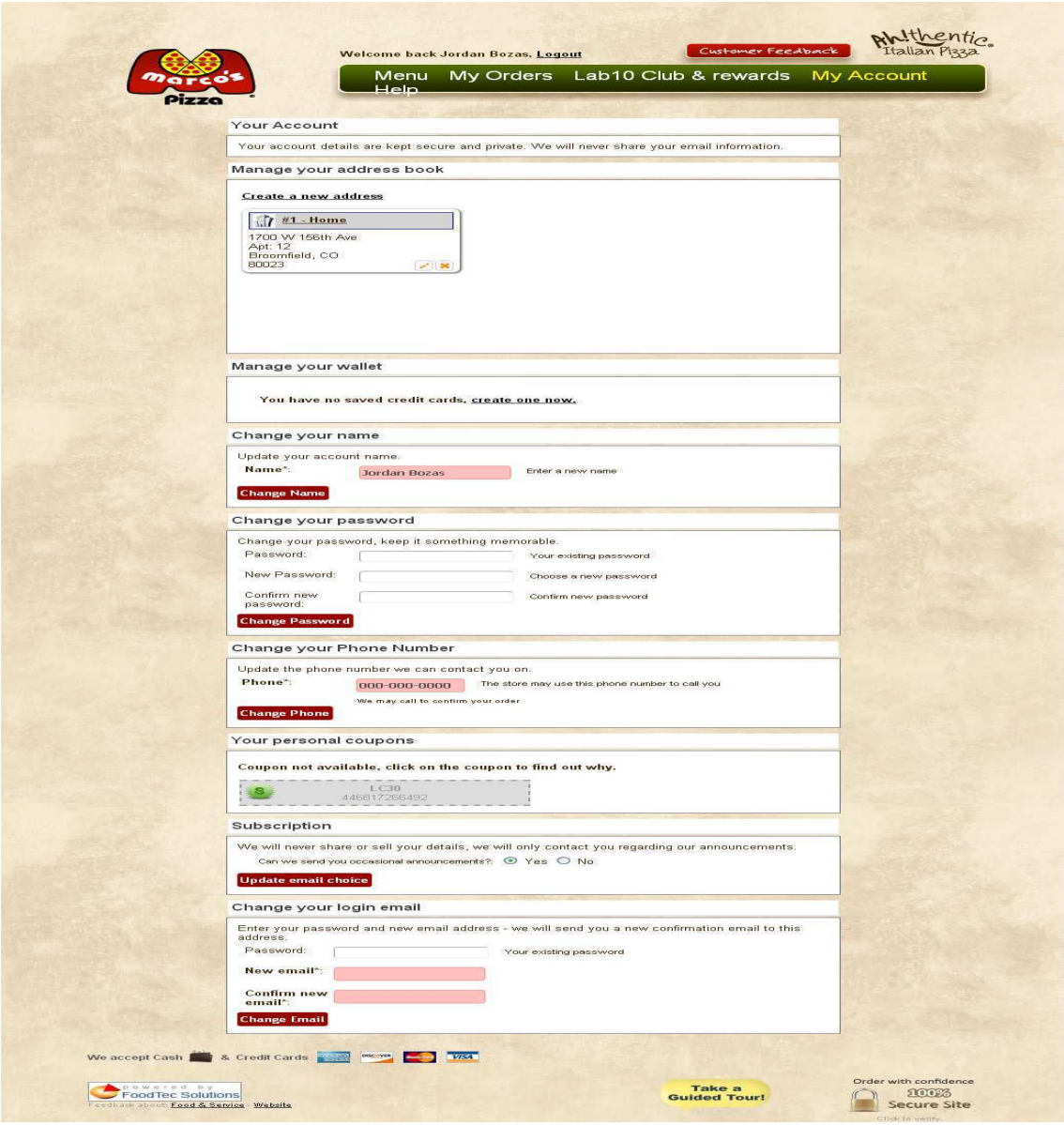
6.Register Page

ΥΠΟΚΕΦΑΛΑΙΟ 1.5

My Account Page

Αυτή είναι η σελίδα από την οποία ο χρήστης μπορεί να διαχειριστεί τον λογαριασμό του. Από εδώ ο χρήστης μπορεί να αποθηκεύσει τις διευθήνσεις τις οποίες χρησιμοποιεί για να παραγγείλει, να αποθηκεύσει τις πιστωτικές κάρτες που χρησιμοποιεί, να τροποποιήσει το όνομά του, να αλλάξει τον κωδικό του, να αλλάξει το τηλέφωνό του καθώς και το email του.

Επίσης μπορεί να ενημερωθεί για τα κουπόνια που διαθέτει και να σταματήσει να λαμβάνει ενημερώσεις από το newsletter.



7.My Account Page

ΥΠΟΚΕΦΑΛΑΙΟ 1.6

My Orders Page

Ο χρήστης μπορεί από αυτή τη σελίδα να δει τις προηγούμενες παραγγελίες του και αν θέλει να πραγματοποιήσει ξανά την ίδια παραγγελία. Έτσι μπορεί να έχει ένα ιστορικό και να εκτυπώσει παλαιότερες αποδείξεις.

Επίσης μπορεί να παρακολουθεί την εξέλιξη της παραγγελίας του και να ενημερώνεται αν είναι έτοιμη και αν είναι στον δρόμο.

Τέλος ο χρήστης μπορεί να βαθμολογήσει τα αντικείμενα πο έχει παραγγείλει αξιολογώντας την ποιότητα τους.

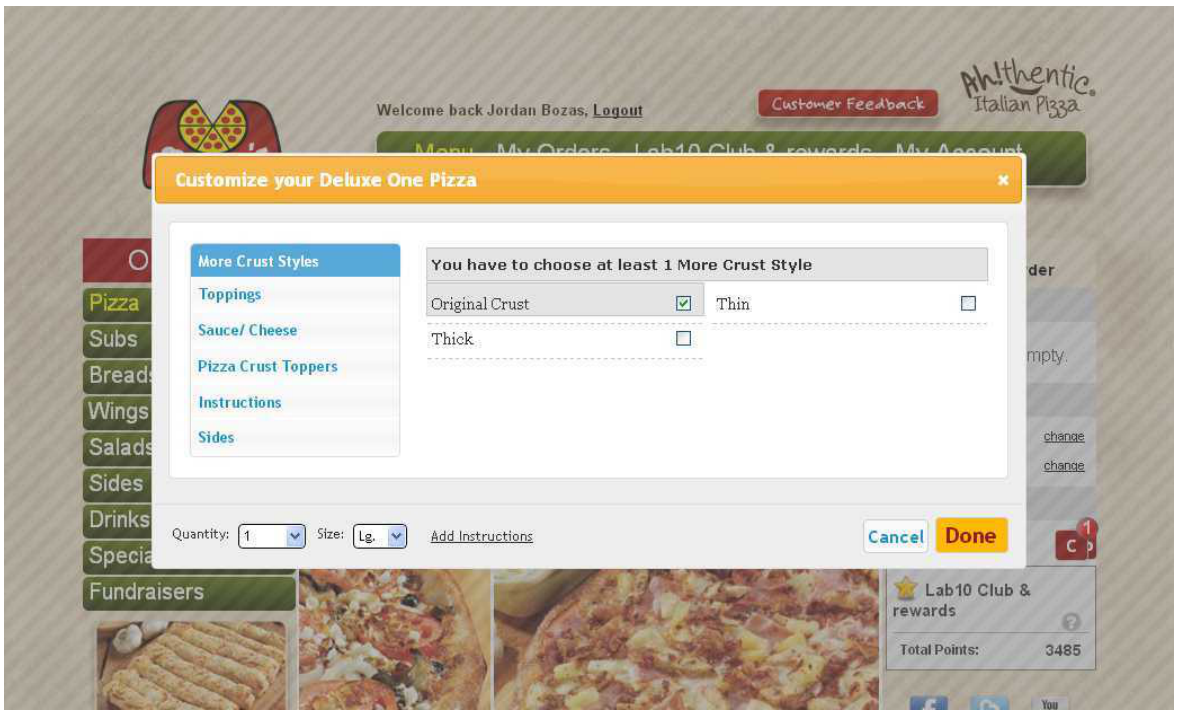
8.My Orders Page

ΥΠΟΚΕΦΑΛΑΙΟ 1.7

Adding Items

Όταν ο χρήστης επιλέξει ένα αντικείμενο για να το προσθέσει στην παραγγελία του θα εμφανιστεί ένα dialog για να μπορέσει ο χρήστης να τροποποιήσει το αντικείμενο έτσι όπως θέλει εκείνος. Μπορεί να επιλέξει από τις κατηγορίες των υλικών και να προσθέσει-αφαιρέσει υλικά. Μπορεί επίσης να ορίσει την ποσότητα και το μέγεθος του αντικειμένου.

Αφότου επιλέξει αυτά που θέλει μπορεί να προσθέσει το αντικείμενο στο καλάθι αγορών του και να συνεχίσει την παραγγελία του. Το αντικείμενο που επέλεξε θα εμφανιστεί στην δεξιά column.



9.Adding Item

ΥΠΟΚΕΦΑΛΑΙΟ 1.8

Checkout Process

Η διαδικασία του Checkout αποτελείται από τρία βήματα.

Order Details

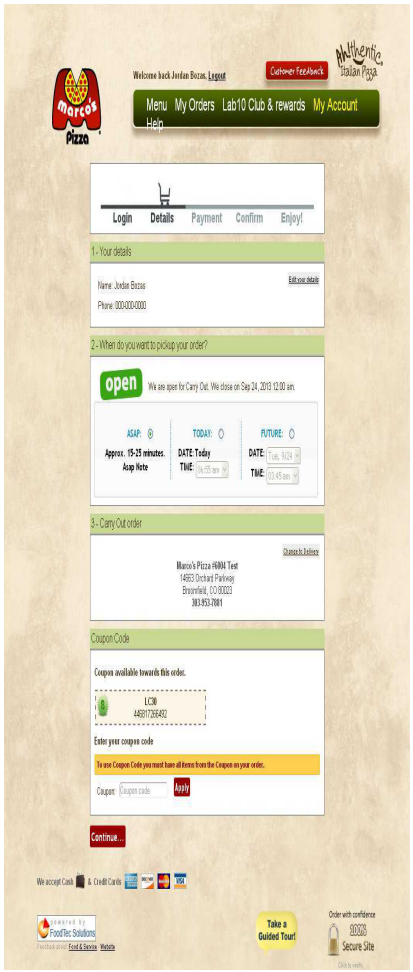
Στο πρώτο βήμα ο χρήστης εισάγει τα στοιχεία του αν δεν είναι logged in, επιλέγει πότε θέλει έτοιμη την παραγγελία του, ορίζει την διεύθυνση και επίσης μπορεί να προσθέσει κάποιο κουπόνι που διαθέτει.
Μόλις έχει συμπληρώσει όλα τα στοιχεία, στη συνέχεια μπορεί να προχωρήσει στο δεύτερο βήμα.

Order Payment

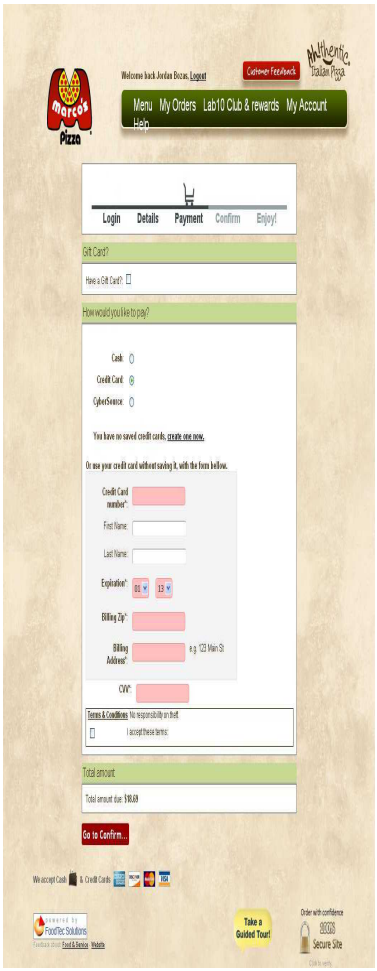
Ο χρήστης μπορεί να επιλέξει το τρόπο πληρωμής που επιθυμεί. Μπορεί να πληρώσει είτε με μετρητα (Cash), είτε με πιστωτική κάρτα (Credit Card), είτε να χρησιμοποιήσει κάποια κάρτα δώρου (Gift Card)

Confirm Order

Ο χρήστης στο τελευταίο βήμα βλέπει μία περίληψη της παραγγελίας του και μπορεί να αλλάξει ότι επιθυμεί από την παραγγελία του. Όταν είναι έτοιμος μπορεί να βάλει την παραγγελία του.



10.Checkout Step



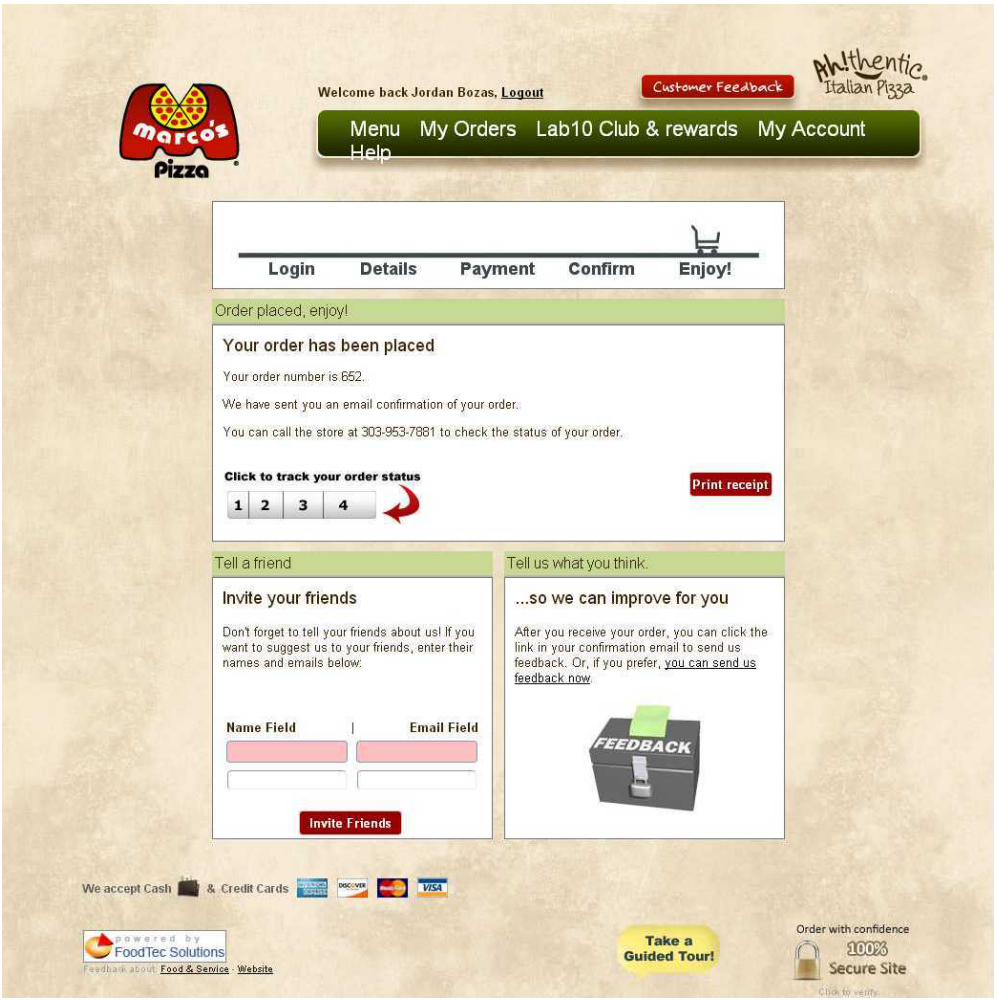
11.Checkout Step-2

12.Checkout Step-3

ΥΠΟΚΕΦΑΛΑΙΟ 1.9

Order Information

Αφού έχει υποβάλει την παραγγελία ο χρήστης εμφανίζεται ένα ενημερωτικό μήνυμα με τον αριθμό της παραγγελίας του.
Μπορεί να επιλέξει τον σύνδεσμο για να κατευθυνθεί στη σελίδα My Orders απ'όπου μπορεί να παρακολουθήσει την εξέλιξη της παραγγελίας του.
Μπορεί να αφήσει feedback με τυχόν παράπονα και υποδείξεις όπως επίσης μπορεί να προσκαλέσει και φίλους του και να τους προτείνει τη σελίδα.



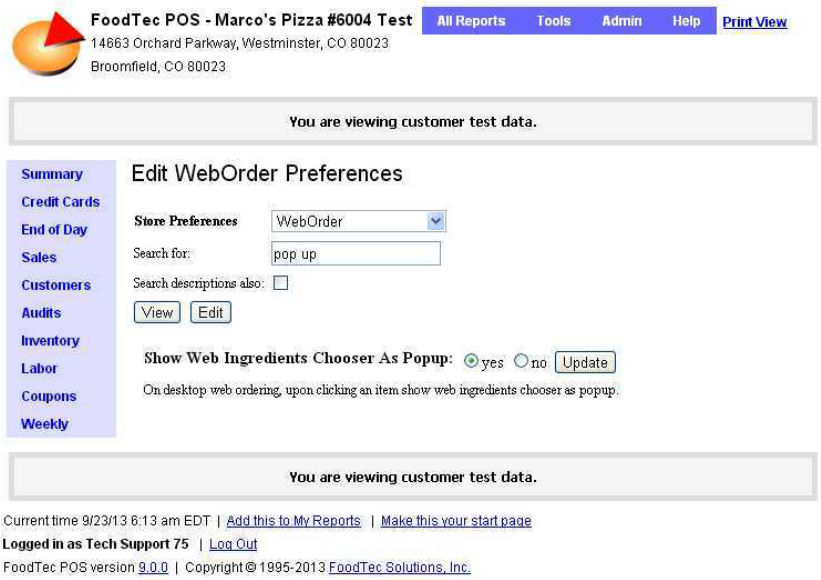
13.Order Information

ΥΠΟΚΕΦΑΛΑΙΟ 1.10

Preferences Editor

Αυτή η σελίδα είναι διαθέσιμη μόνο στον Administrator του συστήματος και δίνει την δυνατότητα στον χρήστη να παραμετροποιήσει τον τρόπο που συμπεριφέρεται το σύστημα.

Από εδώ μπορούν να επιρεαστούν πάρα πολλά preferences που είναι υπεύθυνα για το πως συμπεριφέρεται το σύστημα. Χωρίζονται σε κατηγορίες και ο χρήστης μπορεί δυναμικά να τα αλλάξει.



14.Prefs Editor

ΕΠΙΛΟΓΟΣ

Το σύστημα δίνει πολλές επιλογές στον χρήστη. Του δίνει την δυνατότητα να κάνει γρήγορα και εύκολα την παραγγελία του, καθώς και να διαχειριστεί όλες τις πληροφορίες που έχει αποθηκευμένες στον λογαριασμό του.

ΚΕΦΑΛΑΙΟ 2

Παρουσίαση των εργαλείων

ΕΙΣΑΓΩΓΗ

Πολλές, ίσως οι περισσότερες, εφαρμογές λογισμικού σήμερα είναι γραμμένες ως web-based εφαρμογές να τρέχουν σε ένα πρόγραμμα περιήγησης στο Internet. Η αποτελεσματικότητα του ελέγχου αυτών των εφαρμογών ποικίλλει ευρέως μεταξύ των επιχειρήσεων και οργανισμών. Σε μια εποχή άκρως διαδραστική που ανταποκρίνεται στις διαδικασίες λογισμικού, όπου πολλές οργανώσεις χρησιμοποιούν κάποια μορφή Agile μεθοδολογίας, η αυτοματοποίηση της δοκιμής είναι συχνά απαραίτητη για τα έργα λογισμικού. Η αυτοματοποίηση της δοκιμής είναι συχνά η απάντηση.Τα Test αυτοματισμού πραγματοποιούνται χρησιμοποιώντας ένα εργαλείο λογισμικού για να τρέξει επαναλαμβανόμενες δοκιμές κατά την εφαρμογή που πρόκειται να ελεγχθεί. Για τη δοκιμή παλινδρόμησης(regression testing) αυτό προσφέρει αυτή τη δυνατότητα.

Υπάρχουν πολλά πλεονεκτήματα χρησιμοποιώντας την αυτοματοποίηση. Τα περισσότερα σχετίζονται με την επαναληψιμότητα των δοκιμών και τη ταχύτητα με την οποία μπορούν να εκτελεστούν οι δοκιμές. Υπάρχει μια σειρά εμπορικών και ανοικτού κώδικα εργαλείων διαθέσιμα για να βοηθήσει με την ανάπτυξη της αυτοματοποιημένης δοκιμής. Το Selenium είναι ίσως η πιο ευρέως χρησιμοποιούμενη λύση ανοικτού κώδικα.

Τα test αυτοματισμού έχει συγκεκριμένα πλεονεκτήματα για τη βελτίωση της μακροπρόθεσμης αποτελεσματικότητας των διαδικασιών testing μιας ομάδας λογισμικού. Η αυτοματοποίηση testing υποστηρίζει:

- Τις συχνές δοκιμές παλινδρόμησης (Regression Testing)
- Η ταχεία ανατροφοδότηση για τους προγραμματιστές (Rapid Feedback)
- Σχεδόν απεριόριστες επαναλήψεις των δοκιμών ανά περίπτωση εκτέλεσης
- Υποστήριξη για Agile και ακραία μεθοδολογία ανάπτυξης
- Πειθαρχημένη τεκμηρίωση των περιπτώσεων δοκιμής
- Προσαρμοσμένη αναφοράς για ελαττωματικά προϊόντα
- Η εύρεση ελαττωμάτων που χάνονται με χειροκίνητο έλεγχο

Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση του Selenium και συγκεκριμένα του Selenium Web Driver. Αυτό το εργαλείο θα το χρησιμοποιήσουμε για να μπορέσουμε αυτόματα με κώδικα Java να χειριστούμε όλα τα Web Elements των σελίδων του Web Ordering της Foodtec.

Θα περιγράψουμε αναλυτικά τα βασικά στοιχεία και τις μεθόδους που θα χρησιμοποιήσουμε και θα ακολουθήσουν απλά παραδείγματα με εφαρμογή αυτών των τεχνικών και μεθόδων.

Επίσης θα παρουσιαστεί και μία απλή χρήση του JUnit για την σύνταξη των test cases.

ΥΠΟΚΕΦΑΛΑΙΟ 2.1

Selenium

Εισαγωγή στο Selenium

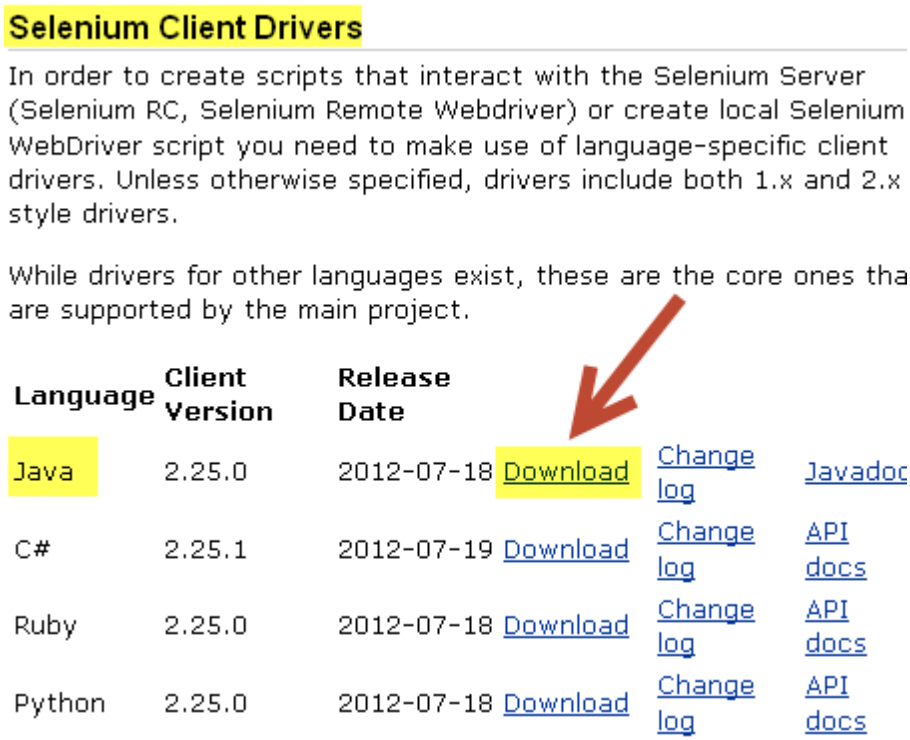
Το Selenium είναι ένα σύνολο από διαφορετικά εργαλεία λογισμικού καθένα με μια διαφορετική προσέγγιση για την υποστήριξη της αυτοματοποιημένης δοκιμής. Οι περισσότεροι QA Μηχανικοί έχουν επικεντρωθεί σε ένα ή δύο εργαλεία που να συμβαδίζουν περισσότερο με τις ανάγκες του έργου τους, ωστόσο, μαθαίνοντας όλα τα εργαλεία θα σας δώσει πολλές διαφορετικές επιλογές για να προσεγγίσετε διαφορετικά προβλήματα αυτοματοποιημένης δοκιμής. Η συνολική σουίτα εργαλείων οδηγεί σε μια πλούσια σειρά από λειτουργίες ελέγχου ειδικά για τις ανάγκες των δοκιμών των web εφαρμογών όλων των τύπων. Οι λειτουργίες αυτές είναι εξαιρετικά ευέλικτες, επιτρέποντας πολλές επιλογές για τον εντοπισμό στοιχείων UI και συγκρίνοντας τα αναμενόμενα αποτελέσματα των δοκιμών με βάση την πραγματική συμπεριφορά της εφαρμογής. Ένα από τα βασικά χαρακτηριστικά του Selenium είναι η υποστήριξη για τη διενέργεια δοκιμών ενός ατόμου σε πολλαπλές πλατφόρμες περιήγησης.

Selenium 2 (aka. Selenium Webdriver)

Το κύριο νέο χαρακτηριστικό στο Selenium 2.0 είναι η ενσωμάτωση του WebDriver API. Το WebDriver έχει σχεδιαστεί για να προσφέρει μια απλούστερη, πιο συνοπτική διεπαφή προγραμματισμού. Το Selenium-WebDriver αναπτύχθηκε για την καλύτερη υποστήριξη δυναμικών ιστοσελίδων, όπου τα στοιχεία μιας σελίδας μπορεί να αλλάξουν χωρίς η ίδια σελίδα να ξαναφορτωθεί. Στόχος του WebDriver είναι να παράσχει ένα καλά σχεδιασμένο object-oriented API που παρέχει βελτιωμένη υποστήριξη για τα σύγχρονα προβλήματα testing ενός web-app.

Εγκατάσταση και χρήση του Selenium Web Driver με το Eclipse

- Το πρώτο βήμα είναι να κατεβάσετε το Selenium Client Drivers Java DLLs από τη σελίδα του Selenium: http://seleniumhq.org/download/



Selenium Client Drivers

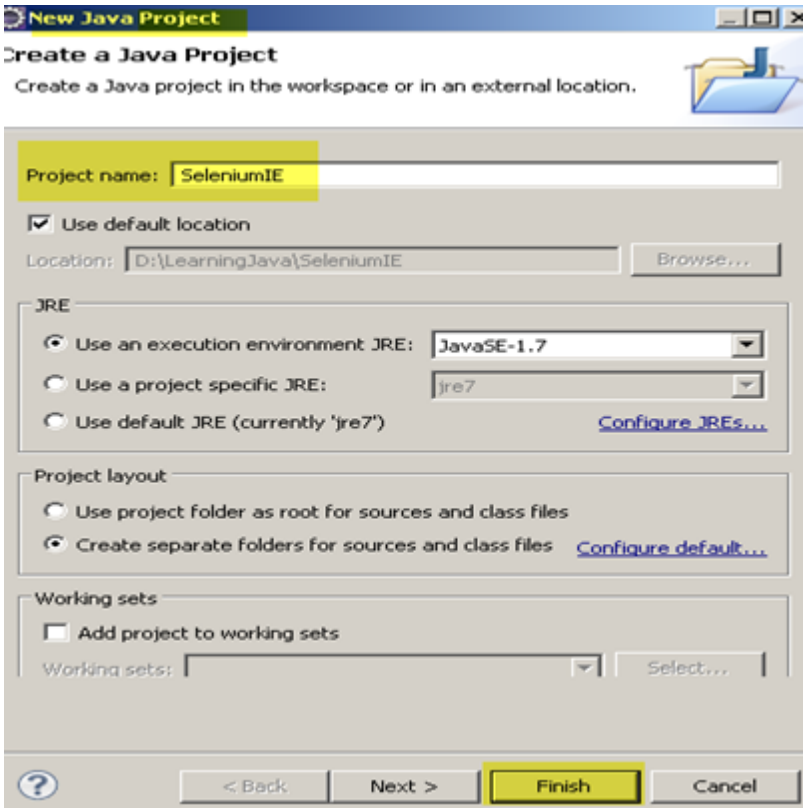
In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote Webdriver) or create local Selenium WebDriver script you need to make use of language-specific client drivers. Unless otherwise specified, drivers include both 1.x and 2.x style drivers.

While drivers for other languages exist, these are the core ones tha are supported by the main project.

Language	Client Version	Release Date			
Java	2.25.0	2012-07-18	Download	Change log	Javadoc
C#	2.25.1	2012-07-19	Download	Change log	API docs
Ruby	2.25.0	2012-07-18	Download	Change log	API docs
Python	2.25.0	2012-07-18	Download	Change log	API docs

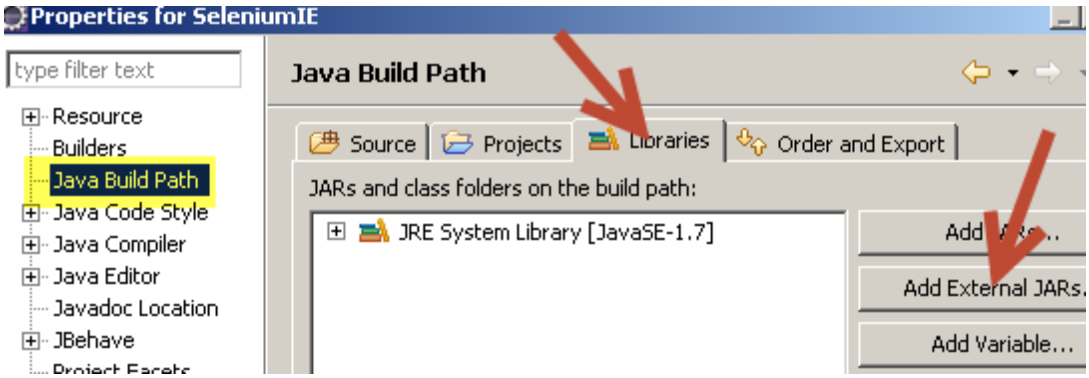
15.Download Selenium

- Αφού έχετε κατεβάσει τα απαραίτητα αρχεία, εξαγάγετε τα αρχεία σε μια τοπική μονάδα δίσκου του υπολογιστή σας.
- Ξεκινήστε το Eclipse και δημιουργήστε ένα νέο Java project.



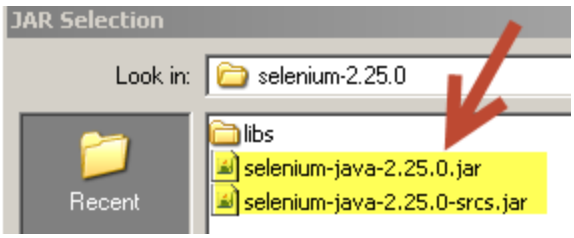
16.Create Eclipse Project

- Πηγαίνετε Propject>Properties menu
- Στα Properties του Project επιλέξτε 'Java Build Path' και στην καρτέλα 'Libraries'

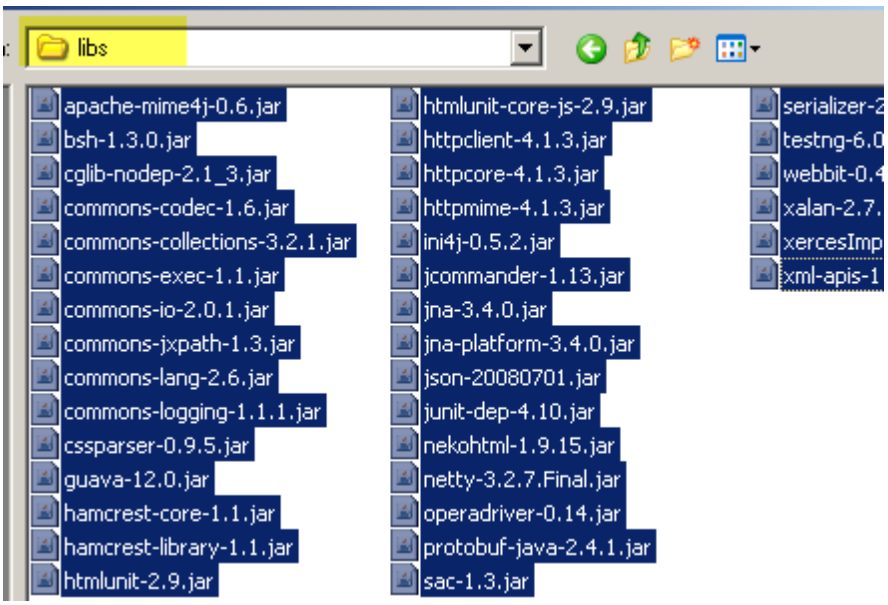


17.Add Libraries

- Επιλέξτε τη διαδρομή όπου αποσυμπιέσατε το Selenium Java Driver
- Προσθέστε όλα τα jars από το selenium-x.xx.x και όλα από τον φάκελο libs



18.Add Jars_1



19.Add Jars_2

- Μόλις έχετε εισάγει όλα τα jars πατήστε **OK** στα Properties για το project και είστε έτοιμοι να ξεκινήσετε

Παρουσίαση του Selenium Web Driver API με παράδειγμα

Ο WebDriver είναι ένα εργαλείο για την αυτοματοποίηση δοκιμών web εφαρμογών, και ιδίως για να εξακριβωθεί ότι λειτουργούν όπως αναμένεται. Στόχος της είναι να παρέχει ένα φιλικό API που είναι εύκολο να το εξερευνήσετε και να κατανοήσετε, το οποίο θα βοηθήσει να κάνετε τις δοκιμές σας πιο εύκολο να διαβαστούν και να διατηρηθούν. Δεν είναι συνδεδεμένο με κάποιο συγκεκριμένο πλαισίοο δοκιμής, έτσι ώστε να μπορεί να χρησιμοποιηθεί εξίσου καλά δοκιμές μονάδας (Unit Testing) ή από μία απλή main μέθοδο. Αυτή η ενότητα παρουσιάζει το API WebDriver και βοηθά να ξεκινήσετε και να εξοικειωθείται με αυτό. Ξεκινήστε με τη δημιουργία ενός project WebDriver αν δεν το έχετε ήδη.

Μόλις το project σας έχει δημιουργηθεί, μπορείτε να δείτε ότι ο WebDriver ενεργεί ακριβώς όπως κάθε κανονική βιβλιοθήκη: είναι εντελώς αυτόνομο, και συνήθως δεν χρειάζεται να θυμάστε να αρχίσετε οποιεσδήποτε επιπρόσθετες διαδικασίες ή να τρέξετε όλα τα προγράμματα εγκατάστασης πριν από τη χρήση.

Τώρα είστε έτοιμοι να γράψετε κώδικα. Ένας εύκολος τρόπος για να ξεκινήσετε είναι αυτό το παράδειγμα, το οποίο ψάχνει για τον όρο "Cheese" στο Google και στη συνέχεια εξάγει τον τίτλο της σελίδας ως αποτέλεσμα στην κονσόλα.

```
package org.openqa.selenium.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.WebDriverWait;

public class Selenium2Example  {
   public static void main(String[] args) {
       // Create a new instance of the Firefox driver
       // Notice that the remainder of the code relies on the interface,
       // not the implementation.
       WebDriver driver = new FirefoxDriver();

       // And now use this to visit Google
       driver.get("http://www.google.com");
       // Alternatively the same thing can be done like this
       // driver.navigate().to("http://www.google.com");

       // Find the text input element by its name
       WebElement element = driver.findElement(By.name("q"));
```

```
        // Enter something to search for
        element.sendKeys("Cheese!");

        // Now submit the form. WebDriver will find the form for us from the element
        element.submit();

        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Google's search is rendered dynamically with JavaScript.
        // Wait for the page to load, timeout after 10 seconds
        (new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>() {
            public Boolean apply(WebDriver d) {
                return d.getTitle().toLowerCase().startsWith("cheese!");
            }
        });

        // Should see: "cheese! - Google Search"
        System.out.println("Page title is: " + driver.getTitle());

        //Close the browser
        driver.quit();
    }
}
```

Selenium-WebDriver API Commands and Operations

Πλοηγηθείτε σε μια σελίδα

Το πρώτο πράγμα που είναι πιθανό να θέλετε να κάνετε με τον WebDriver είναι να πλοηγηθείτε σε μια σελίδα. Ο κανονικός τρόπος για να γίνει αυτό είναι με την κλήση της "get":

```
driver.get("http://www.google.com");
```

Βρίσκοντας UI Αντικείμενα (Web Elements)

Για να εντοπίσετε στοιχεία UI μπορείτε είτε χρησιμοποιώντας το instance του WebDriver είτε με τη χρήση του WebElement. Κάθε ένα από αυτά τα bindings θα εμφανίσει μία Find Element ή Find Elements μέθοδο. Το πρώτο επιστρέφει ένα αντικείμενο WebElement αλλιώς πετάει ένα exception. Το δεύτερο επιστρέφει μια λίστα WebElements ή μπορεί να επιστρέψει μια κενή λίστα, αν δεν υφίστανται στοιχεία DOM που να ταιριάζουν με το ερώτημα.

Η μέθοδος “Find” δέχεται παράμετρο ή ερώτημα που ονομάζεται “By”.

By ID

Αυτός είναι ο πιο αποτελεσματικός και σωστός τρόπος για να εντοπίσουμε ένα element. Πολλές φορές βέβαια θα συναντήσουμε elements τα οποία δεν έχουν id.

```
<div id="coolestWidgetEvah">...</div>

WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
```

By Class Name

Η κλάση του DOM element είναι αρκετά πρακτίκη γιατί μπορεί να μας επιστρέψει μία λίστα με όλα τα στοιχεία που ανήκουν σε αυτήν την κλάση.

```
<div class="cheese"><span>Cheddar</span></div><div
class="cheese"><span>Gouda</span></div>

List<WebElement> cheeses = driver.findElements(By.className("cheese"));
```

By Tag Name

Χρησιμοποιώντας το tag name του στοιχείου.

```
<iframe src="..."></iframe>

WebElement frame = driver.findElement(By.tagName("iframe"));
```

By Name

Αντίστοιχα με το όνομα του στοιχείου

```
<input name="cheese" type="text"/>

WebElement cheese = driver.findElement(By.name("cheese"));
```

By Link Text

Χρησιμοποιώντας το link text του στοιχείου

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>

WebElement cheese = driver.findElement(By.linkText("cheese"));
```

By Partial Link Text

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

WebElement cheese = driver.findElement(By.partialLinkText("cheese"));
```

By CSS

Με βάση το CSS του στοιχείου. Αυτό μπορεί να διαφέρει ανάλογα με τον browser. Η αντιστοίχηση γίνεται με βάση τους w3c css selectors.

```
<div id="food"><span class="dairy">milk</span><span class="dairy
aged">cheese</span></div>

WebElement cheese = driver.findElement(By.cssSelector("#food span.dairy.aged"));
```

By XPATH

Σε υψηλό επίπεδο, ο WebDriver χρησιμοποιεί τις native δυνατότητες XPath ενός browser στο μέτρο του δυνατού. Σε αυτά τα προγράμματα περιήγησης που δεν έχουν εγγενή υποστήριξη XPath, έχουμε παράσχει τη δική μας εφαρμογή. Αυτό μπορεί να οδηγήσει σε κάποια μη αναμενόμενη συμπεριφορά αν δεν είστε ενήμεροι για τις διαφορές στις διάφορες μηχανές xpath.

```
<input type="text" name="example" />
<INPUT type="text" name="other" />

List<WebElement> inputs = driver.findElements(By.xpath("//input"));
```

Using JavaScript

Με τη χρήση JavaScript. Μπορεί να χρησιμοποιηθεί εκτελώντας κώδικα JavaScript αρκεί να επιστρέφετε ένα DOM element.

```
WebElement element = (WebElement) ((JavascriptExecutor)driver).executeScript("return $('.cheese')[0]");
```

Βρισκοντας όλα τα inpute elements για κάθε label της σελίδας:

```
List<WebElement> labels = driver.findElements(By.tagName("label"));
List<WebElement> inputs = (List<WebElement>)
((JavascriptExecutor)driver).executeScript(
   "var labels = arguments[0], inputs = []; for (var i=0; i < labels.length; i++){" +
   "inputs.push(document.getElementById(labels[i].getAttribute('for'))); } return inputs;",
labels);
```

User Input - Filling In Forms

Μπορείτε να κάνετε "toggle" την κατάσταση των πλαισίων ελέγχου, και μπορείτε να χρησιμοποιήσετε το "κλικ" για να ρυθμίσετει ένα tag OPTION να επιλεγεί. Επίσης μπορείτε να χειριστείτε τα SELECT tags:

```
WebElement select = driver.findElement(By.tagName("select"));
List<WebElement> allOptions = select.findElements(By.tagName("option"));
for (WebElement option : allOptions) {
   System.out.println(String.format("Value is: %s", option.getAttribute("value")));
   option.click();
}
```

Αυτό θα βρεί το πρώτο "SELECT" στοιχείο στη σελίδα, και θα γυρίσει μέσα από κάθε ένα από τις επιλογές του με τη σειρά, εκτύπωνωντας τα values τους, και επιλέγοντας το καθένα με τη σειρά του. Όπως θα παρατηρήσετε, αυτός δεν είναι ο πιο αποτελεσματικός τρόπος για να αντιμετώπισεται SELECT στοιχεία. Υπάρχουν επιλογές του WebDriver που ονομάζονταιι "Select", η οποία παρέχει χρήσιμες μέθοδους για την αλληλεπίδραση με αυτά.

```
Select select = new Select(driver.findElement(By.tagName("select")));
select.deselectAll();
select.selectByVisibleText("Edam");
```

Αυτό θα απενεργοποιήσει όλες τις επιλογές του Select και θα επιλέξει το Option που περιέχει το Text “Edam”.

Μόλις τελειώσετε τη συμπλήρωση της φόρμας, θέλετε να την υποβάλετε. Ένας τρόπος για να γίνει αυτό θα ήταν να βρείτε το κουμπί "Αποστολή" και να κάνετε κλικ σε αυτό:

```
driver.findElement(By.id("submit")).click();
```

Αλλιώς μπορείτε να χρησιμοποιήσετε την μέθοδο “submit” που προσφέρει το WebDriver. Αν την καλέσετε αυτό θα ακολουθήσει το DOM και θα υποβάλει την φόρμα στην οποία ανήκει το element.

```
element.submit();
```

Μετακίνηση μεταξύ Windows και Πλαίσια

Ορισμένες εφαρμογές web έχουν πολλά πλαίσια ή πολλαπλά παράθυρα. Ο WebDriver υποστηρίζει να κινείται μεταξύ παραθύρων χρησιμοποιώντας τη μέθοδο "switchTo":

```
driver.switchTo().window("windowName");
```

Επίσης μπορείτε να μετακινηθείτε από frame σε frame.

```
driver.switchTo().frame("frameName");
```

Μπορείτε επίσης να μετακινηθείτε σε subframe διαχωρίζοντας το path με τελεία.

```
driver.switchTo().frame("frameName.0.child");
```

Popup Dialogs

Το Selenium δίνει την δυνατότητα να κατευθυνθείτε σε Pop up dialogs. Αφού έχει ενεργοποιηθεί το pop up μπορείτε να έχετε πρόσβαση σε αυτό με την παρακάτω εντολή:

```
Alert alert = driver.switchTo().alert();
```

Αυτή η εντολή θα σας επιστρέψει το ανοιχτό pop up. Με αυτό το αντικείμενο μπορείτε να κάνετε accept, dismiss, να διαβάσετε τα περιεχόμενα ή να γράωετε σε κάποιο prompt.

Navigation: History and Location

Το navigation είναι από τα συμαντικότερα task για να μπορέσετε να χειριστείτε σελίδες. Υπάρχουν διάφορες εντολές για χρησιμοποιήσετε το navigate:

```
driver.navigate().to("http://www.example.com");

driver.navigate().forward();
driver.navigate().back();
```

Drag And Drop

Παρακάτω είναι ένα παράδειγμα που δείχνει πως να χρησιμοποιήσετε το drag n drop

```
WebElement element = driver.findElement(By.name("source"));
WebElement target = driver.findElement(By.name("target"));

(new Actions(driver)).dragAndDrop(element, target).perform();
```

ΥΠΟΚΕΦΑΛΑΙΟ 2.2

Selenium και JUnit

Για να μπορέσουμε να δημιουργήσουμε Test Cases και να έχουμε ένα περιβάλλον που θα μπορούμε να τρέξουμε μαζικά τα test cases μας θα χρησιμοποιήσουμε το γνωστό εργαλείο JUnit έτσι ώστε να συντάξουμε και να διαχειριστούμε αυτά τα tests.

Παρακάτω παρουσιάζεται ένα παράδειγμα με την δομή ενός test:

```
package com.btmatthews.selenium.junit4.runner.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.openqa.selenium.WebDriver;

import com.btmatthews.selenium.junit4.runner.SeleniumJUnit4ClassRunner;
import com.btmatthews.selenium.junit4.runner.SeleniumWebDriver;
import com.btmatthews.selenium.junit4.runner.WebDriverConfiguration;

/**
 * Unit tests for the {@link WebDriverConfiguration} configuration style.
 *
 * @author <a href="mailto:brian@btmatthews.com">Brian Thomas
Matthews</a>
 * @since 1.0.0
 */
@RunWith(SeleniumJUnit4ClassRunner.class)
@WebDriverConfiguration()
public final class TestWebDriver {

        /**
         * The object used to start/stop the web browser used for
testing.
         */
        @SeleniumWebDriver
        private WebDriver webDriver;

    /**
     * The name of the browser being used for the test.
     */
        @SeleniumBrowser
        private String browserName;

        /**
         * Verify that the test runner injected the web driver.
         */
        @Test
        public void testInjection() {
```

```
                assertNotNull(webDriver);
                assertNotNull(browserName);
                assertEquals("HtmlUnitDriver", browserName);
        }

        /**
         * Verify that we can navigate to the Google home page.
         */
        @Test
        public void testHomePage() {
                webDriver.navigate().to("http://www.google.com");
                assertEquals("Google", webDriver.getTitle());
        }
}
```

Χρησιμοποιώντας annotations μπορούμε να ορίσουμε την σουίτα και το configuration που θα ανήκει η test κλάση που θέλουμε.

@RunWith

@WebDriverConfiguration

Στη συνέχεια ορίζουμε τα Tests μας και προσθέτουμε τον κωδικα του Selenium στα tests μας.

@Test

ΕΠΙΛΟΓΟΣ

Τα εργαλεία μας δίνουν όλες τις δυνατότητες για να δομήσουμε και να αυτοματοποιήσουμε τις διαδικασίες ούτως ώστε να επιτευχθεί η αυτοματοποίηση των ελέγχων.

Με αυτό τον τρόπο μπορούμε να εκμεταλλευτούμε την αυτοματοποίηση για την γρήγορη εκτέλεση των ελέγχων βελτιστοποιώντας τον κύκλο ζωής μίας νέας έκδοσης.

ΚΕΦΑΛΑΙΟ 3

Ανάπτυξη των Test Cases

ΕΙΣΑΓΩΓΗ

Για να έχει αξία μία αυτοματοποιημένη διαδικασία ελέγχου, χρειάζεται να υπάρχουν τα απαραίτητα test cases τα οποία καλύπτουν όλες τις περιπτώσεις που πρέπει να ελεγχθούν.

Αυτό το κομμάτι είναι ιδιαίτερα σημαντικό. Από τη στιγμή που θα είναι επιτυχημένη η εκτέλεση των test cases είμαστε βέβαιοι ότι η λειτουργικότητα του συστήματός μας είναι σωστή και ότι ο τελικός χρήστης θα πάρει τα σωστά αποτελέσματα.

Για να δομήσουμε κατάλληλα τα test cases θα ακολουθήσουμε στο σχεδιασμό μας δύο βασικούς ελέγχους. Την ακεραιότητα των στοιχείων τα οποία παρουσιάζονται στον χρήστη και τον έλεγχο για σωστά αποτελέσματα.

Παρακάτω θα περιγραφούν αναλυτικά οι περιπτώσεις που είναι απαραίτητς να καλυφθούν για να ελέγξουμε την ακεραιότητα του τελικού συστήματος ως προς την εμφάνιση και την λειτουργία του.

ΥΠΟΚΕΦΑΛΑΙΟ 3.1

Ελέγχοντας την ακεραιότητα του συστήματος

Η συγκεκριμένη οπτική γωνία θα μας εξασφαλίσει ότι όλα τα elements τα οποία υπάρχουν στην σελίδα μας θα είναι διαθέσιμα στον χρήστη. Είναι σημαντικό να ελέγχεται το σύστημα με αυτόν τον τρόπο, μιας και μετά από οποιαδήποτε αλλαγή μπορεί η αλλαγή του κώδικα να προκαλέσει ένα element να μην εμφανίζεται σωστά ή και καθόλου.

Θα προσπαθήσουμε να ελέγξουμε κάθε φόρμα και input field έτσι ώστε να είμαστε σίγουροι ότι θα έχουμε το επιθυμητό αποτέλεσμα.

- Basic Layout
    - Page Layout (Correct Divs available)
        - Intro Page
            - Select Delivery
            - Both Address and log in divs are available
            - Select Carry Out
            - Start your order is available
        - Menu Page
            - Menu links available
            - Menu categories available
            - Your order div available
        - Login Page
            - Register Available
            - Login Available
            - Facebook login available
    - Form Layout (Correct input fields available)
        - Intro Address
            - Error if no values
            - Correct redirect
        - Intro Log in
            - Error if no values
            - Correct redirect
        - Intro Carry Out
            - Start you order-Correct redirect
- Wizards
    - Pop Up Wizards
    - Upsell Items
- Account
    - Addresses

    - CC
    - Phone
    - Email
    - Password
- Payment Checkout
    - CC payment
    - Tips
    - Cash

Όπως αναφέρεται στα παραπάνω θέλουμε να ελέγξουμε για κάθε σελίδα την ύπαρξη των σωστών elements και την σωστή εναλλαγή σελίδων ύστερα από ένα submit μίας φόρμας.

Πρέπει να είμαστε σίγουροι μετά από την εκτέλεση ενός τέτοιου πακέτου tests ότι ο τελικός χρήστης δεν θα έχει κανένα πρόβλημα ώστε να εισάγει τα δεδομένα που επιθυμεί και ότι θα πάρει τα σωστά αποτελέσματα ύστερα από την εκτέλεση της κάθε εντολής που θα επιλέξει. Από αυτό εξαρτάται η αξιοπιστία και η ποιότητα του προϊόντος το οποίο θα διατεθεί στους πελάτες.

ΥΠΟΚΕΦΑΛΑΙΟ 3.2

Ελέγχοντας την σωστή ροή του συστήματος

Το σύστημα πρέπει να έχει ένα συγκεκριμένο τρόπο λειτουργίας ο οποίος έχει προκαθοριστεί κατά την σχεδίαση. Ύστερα από μία αλλαγή στο σημείο που θέλουμε να προσθέσουμε μία νέα λειτουργία, πρέπει να είμαστε βέβαιοι ότι δεν έχουμε χαλάσει κάποια παλαιότερη λειτουργία.

Για αυτόν τον σκοπό έχουμε δημιουργήσει κάποια test cases έτσι ώστε να είμαστε βέβαιοι ότι θα τα τρέξουμε σε κάθε regression test και ότι θα έχουμε τα σωστά αποτελέσματα.

Τα ακόλουθα tests είναι αυτά τα οποία τρέχουν από την ομάδα των QA testers της FoodTec σε κάθε regression test ώστε να εξασφαλιστεί η ορθή λειτουργία του συστήματος.

Μέχρι πρότινος η εκτέλεση αυτών των tests γινόταν χειροκίνητα χωρίς την χρήση κάποιου εργαλείου όπως το Selenium.

Για καλύτερη απόδοση της περιγραφής θα παρουσιαστεί το πρωτότυπο στα Αγγλικά

Title	Setup	Test	Results
Do not allow unregistered User To Order - Delivery Order	- Make sure preference "Allow Unregistered users to Order" is disabled - Make sure preference "Allow Unverified users to Order" is enabled - Do not Log In	1. Enter Web Ordering 2. Select Delivery Order 3. Choose some items and press "Checkout"	Step 3: - Verify that you cannot continue with placing your order. The only actions allowed is "Login" and "Register Now"
Do not allow unregistered User To Order – PickUp Order	- Make sure preference "Allow Unregistered users to Order" is disabled - -Make sure preference "Allow	1. Enter Web Ordering 2. Select PickUp Order 3. Choose some items and press "Checkout"	Step 3: - Verify that you cannot continue with placing your order. The only actions allowed is "Login" and "Register Now"

	Unverified users to Order" is enabled - Do not Log In		
Unregistered User -Do not allow Cash Payment	- Preference "Allow Unregistered users to Order" = true - Preference "Allow Unverified users to Order" = true - Do not Log In	1. Change the Preference: "Allow web cash payments" = "Never" 2. Go to Web Ordering, Select Delivery Order, Choose some items. 3. Press "Check Out" and select "Order right now". Fill in the required information and hit "Continue". Verify you cannot pay with Cash, there is not such an option 4. Select Pickup Order 5. Change the Preference: "Allow web cash payments" = "Registered Customer" 6. Go to Payment page again and verify there is not an option for paying with Cash 7. Change the Preference: "Allow web cash payements" = "Customers With existing Orders" 8. Go to Payment page again and verify there is not an option for paying with Cash	Verify that in any case you cannot proceed with Placing your Order paying with Cash
Verify that in any case you cannot proceed with Placing your Order paying with Cash	- Preference "Allow Unregistered users to Order" = true - Preference "Allow Unverified users to Order" = true - Preference "Allow web cash payments" = always - Edit Store Hours. Set the hours so that Web Ordering is Open - Do not Log In	1. Go to Web Ordering 2. Select Delivery Order 3. Choose some items and press "Checkout" 4. Press "Order Right Now" 5. Fill in the required information, choose to pay with cash and in the "When do you want the order delivered?" section choose Now (asap) 6. Press the "Next Step" button. Mark your order number and your order amount 7. Press "Place Order"	Step 7: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Unregistered User – Allow Cash Payment – Pickup Order	- Preference "Allow Unregistered users to Order" = true - Preference "Allow Unverified users to Order" = true	1. Go to Web Ordering 2. Select Pickup Order 3. Choose some items and press "Checkout" 4. Press "Order Right Now" 5. Fill in the required information,	Step 7: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports

	- Preference "Allow web cash payments" = always - Edit Store Hours. Set the hours so that Web Ordering is Open - Do not Log In	choose to pay with cash and in the "When do you want the order ready?" section choose Now (asap) 6. Press the "Next Step" button. Mark your order number and your order amount 7. Press "Place Order"	-> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Unregistered User – Do Not Allow Credit Card Payment	- Preference "Allow Credit Card Payments" = false - Preference "Allow Unregistered users to Order" = true - Preference "Allow Unverified users to Order" = true - Do not Log In	1. Go to Web Ordering 2. Select Pickup Order 3. Choose some items and press "Checkout" 4. Press "Order Right Now" 5. Verify there is no option for paying with Credit Card 6. Repeat steps 2-5 but select Delivery Order	Verify there is no option for paying with Credit Card
Unregistered User – Allow Credit Card Payment	- Preference "Allow Credit Card Payments" = true - Preference "Allow Unregistered users to Order" = true - Preference "Allow Unverified users to Order" = true - You will need a Test Credit Card - Do not Log In - Make sure the "all pickups paid at store" preference is false	1. Go to Web Ordering 2. Select Pickup Order 3. Choose some items and press "Checkout" 4. Press "Order Right Now" 5. Fill in the required information, choose to pay with Credit Card. 6. Fill in the required information from your Test Credit Card 7. Press the "Next Step" button. Mark your order number and your order amount 8. Press "Place Order" 9. Repeat steps 2-8 but select Delivery Order	- Go to POS, find your orders from "Find" page -> Delivery/Pickup tab and verify its status (correct items, amount and due time). - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Register		1. Go to Web Ordering, My Account page 2. Press "Register" 3. Fill in a wrong Captcha value 4. Press "Register" 5. Fill in the correct Captcha value and press "Register"	Step 4: - Verify error message about the Captcha value, you cannot proceed with Registering Step 5: -Verify that you automatically logged in. -Verify you received

			a Verification Email but do not follow the link
Do not allow unverified User To Order	- Preference "Allow Unverified Users to Order" = false - Log In	1. Select some Items, Choose Delivery Order 2. Press "Check Out" 3. Verify there you cannot proceed with placing you Order 4. Repeat the steps 1,2 and 3 but choose PickUp Order	Verify that in any case you cannot proceed with Placing your Order
Unverified User -Do not allow Cash Payment	- Preference "Allow Unverified Customers to Order" = true - Log In	1. Change the Preference: "Allow web cash payments" = "Never" 2. Go to Web Ordering, Select Delivery Order, Choose some items. 3. Press "Check Out". 4.When do you want the order delivered? Choose ASAP 5.Press Continue.Verify that only Credit card payment option is available.(CyberSource should also be available if you have license for it) 6. Follow steps 2 and 3 but select Pickup Order 7. Change the Preference: "Allow web cash payments" = "Customers With existing Orders" 8. Follow steps 2, 3 and 4	Verify that in any case you cannot proceed with Placing your Order with cash payment
Unverified User - Allow Cash Payment	- Preference "Allow Unverified users to Order" = true - Log In	1. Change the Preference: "Allow web cash payments" = "Registered Customers" 2. Go to Web Ordering, Select Delivery Order, Choose some items. 3. Press "Check Out". Fill in the required information, select Asap and press "Place Order" 4. Follow steps 2 and 3 but select Pickup Order 5. Change the Preference: "Allow web cash payments" = "Customers With existing Orders" 6. Follow steps 2, 3 and 4	Step 6: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Open Orders : Verify your order is there
Unverified User - Allow	- Preference "Allow Unverified users to Order" = true	1. Go to Web Ordering 2. Select Pickup Order 3. Choose some items and press	Step 8: - Go to Web Reports -> All Reports ->

Credit Card Payment	- Preference "Allow Credit Card Payments" = true - You will need a Test Credit Card - Log In	"Checkout" 4. Fill in the required information, choose to pay with Credit Card. 5. Fill in the required information from your Test Credit Card 6. Press the "Next Step" button. Mark your order number and your order amount 7. Press "Place Order" 8. Repeat steps 2-8 but select Delivery Order	Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Unverified User – ReOrder	- Preference "Allow Unverified users to Order" = true - Log In	1. Go to Web Ordering 2. Select Pickup Order 3. From 'My Orders' page select a previous order and press "ReOrder" 4. Fill in the required information 5. Press the "Next Step" button. Mark your order number and your order amount 6. Press "Place Order" 7. Repeat steps 2-8 but select Delivery Order	Step 7: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Pickup Order	- Log In	1. Go to Web Ordering and start a pickup order 2. Select a Pizza and customize it 3. Choose a couple of default/non default ingredients, choose a side and set a qualifier (light, Xxtra...). Test also enforced ingredient choices. 4. Verify that the order tree is correct and place the order 5. Repeat the steps above for mobile web ordering	Steps 3: Verify that on mobile web ordering you are redirected directly to the ingredients page of the first enforced choice. If the item has another enforced choice you will should be redirected to its ingredients page etc. Step 4: From Confirmed page press the back button and verify that no error is thrown Steps 4,5: - Go to POS, find your order from "Find" page -> Pickup tab and verify its status (correct items, amount and due time). - Go to Web Reports

			-> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there - Verify you received an Email with your Order's Details and the information is correct.
Delivery Order	- Log In	1. Go to Web Ordering and start a delivery order 2. Select a Pizza and customize it 3. Choose a couple of default/non default ingredients, choose a side and set a qualifier (light, Xxtra...). Test also enforced ingredient choices. 4. Verify that the order tree is correct and place the order 5. Repeat the steps above for mobile web ordering	Step 4: From Confirmed page press the back button and verify that no error is thrown Steps 4,5: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
ReOrder	- Log In	1. Go to My Orders 2. Select an Order and press ReOrder 3. Repeat the steps above for mobile web ordering	Step 2: - Go to Web Reports -> All Reports -> Journal : Verify correct printing - Go to All Reports -> Web Orders : Verify your order is there - Go to All Reports -> Web Open Orders : Verify your order is there
Save Delivery Addresses	- Log In	1. Go to Web Ordering, select some items, choose Delivery and hit "CheckOut"	Verify no broken layout or malfunction of the Delivery form.

		2. Create a delivery address, check 'Save your Address' and place your Order 3. Repeat step 1 and 2 with a new delivery address 4. Place 2 delivery Orders using the two delivery addresses you created.	Step 3: - Go to Checkout Page again, verify your 2 addresses are saved correctly.
Edit Delivery Addresses	-Log in	1. Go to CheckOut Page with a delivery Order 2. Select one of your saved addresses and press "Delete" 3. Relog in web ordering and go to checkout page again 4. Select another saved Address and press "Edit" 5. Modify the address and continue with placing your Order 6. Go to Checkout Page again with a delivery Order and press "New"	Verify no broken layout or malfunction of the Delivery form. Step 3: - Verify the deleted address is not there Step 4: - Verify the correct fields are enabled for editing your address Step 6: - All fields should be cleared and enabled. You should be able to fill in a new delivery Address
Allow Address Out of Delivery	- Log In - Preference "Allow addresses out of delivery" = true	1. Go to Checkout Page with a delivery Order 2. Fill in a new Address out of delivery Area 3. Press "Next Step" 4. Place your Order	Step 3: - Verify System will prompt a warning message that your Address is out of the store's delivery area. A checkbox will appear as well for "Double Check your address"
Do not Allow Address Out of Delivery	- Log In - Preference "Allow addresses out of delivery" = false	1. Go to Checkout Page with a delivery Order 2. Fill in a new Address out of delivery Area 3. Press "Next Step"	Step 3: - Verify System will not let you continue with placing your order with the error message that "We do not deliver to that address"
Save Credit Cards	- Log In - You will need a Test Credit Card - Preference "Allow web customers to	1. Go to CheckOut Page 2. Choose to pay with Credit Card 3. Fill in the Credit Card Information, check "Save Credit	Verify no broken layout or malfunction of the the Credit Card form. Step 5: - Go to

	store their credit cards" = true	Card Info" 4. Continue with placing your Order 5. Follow steps 1-4 but with a different Credit Card 6. Select one, fill in the CVV number and place your Order	Checkout Page and Verify your saved Credit Cards are saved Step 6: - Go to POS, find your Order and verify the payment information.
Edit Credit Cards	- Log In - You will need a Test Credit Card - Preference "Allow web customers to store their credit cards" = true - Preference "Web Prompt For Billing Zip" = true - Preference "Web Prompt For Billing Addr" = true	1. Go to "My Account" page 2. Press the "x" button to delete one of your saved Credit Cards 3. Relog to web Ordering and go to checkout page 4. Select another saved Credit Card and press "Edit", fill in the required fields with another card's information 5. Place your Order	Verify no broken layout or malfunction of the the Credit Card form. Step 5: - Verify the deleted Credit Card is not present Step 6: - Go to POS, find your Order and verify the edited payment information.
Do not Allow Saved Credit Cards	- Log In - Preference "Allow web customers to store their credit cards" = false	1. Go to CheckOut Page 2. Choose to pay with Credit Card	Step 2: - Verify you cannot see your saved Credit Cards and the "Save your Credit Card" checkbox is not present
Deferred Order	- Preference "max deferred days" = 10 - Preference "print web defers" = true	1. Go to CheckOut Page 2. From the Date Time chooser section, select a future date and time. You can defer an order for later today or for a future day 3. Press "Next Step" 4. Go Back to the checkout Page, and choose a date and time where the store is open 5. Proceed with placing your order	Step 2: - Verify that the available defer dates are 10 Step 3: - You shouldn't be able to proceed. Verify error message "Store will be closed then" Step 5: - Go to Web Reports -> Journal Report, verify the Order printed to the kitchen
Item Editing	- Preference "show qualifiers" = true - Preference "always show item editor" = false	1. Go to Web Ordering 2. Select a Pizza and customize it 3. Choose a couple of ingredients and set a qualifier (light, Xxtra...) 4. Write some text in the	Step 2: - Verify that after item selection, item Editor did not show Step 3:

	- Preference "Allow customer instructions" = true	Instructions field 5. Proceed with placing your Order 6. Do steps 1 - 4 but instead of placing the order open a new tab to see the edit page of the item 7. Clear the order from the first tab and add an ingredient from the edit tab.	- Verify that the Order Tree is updated with the changes correctly - Verify that only one Qualifier panel is visible at a time, and UI accordion is functional for all browsers Step 4: - Verify that there is an instructions field - Verify that the Order tree is updated dynamically with the Instructions correctly Step 5: - Go to POS, find your Order and verify the Order's Status (Items, qualifiers, instructions) Step 7: - Verify that you are redirected to start page without any problem
Minimum delivery amount	- Preference "min delivery amount" = $15	1. Go to Web Ordering 2. Select some items that do not exceed the value of $15 and press checkout 3. Change the Order Type to Pickup and press "CheckOut" 4. Go Back to "Order Now" page and change the Order Type to delivery. 5. Add some more items to your order to exceed the $15 6. Continue with placing your order	Step 2: - Verify warning message about the minimum delivery amount, you cannot proceed on checkout Step 3: - You should be able to proceed since the Order Type is Pickup Step 6: - Go to POS, find your Order and verify the payment information
Edit My Account Information	- Log In	1. Go to My Account Page 2. Change your Password 3. Relog with your new password 4. Change your Phone Number	Step 2: - Try not to put correct values with the first time, insert different new

		5. Change your Email 6. Relog with your new Email	Password and confirm password or enter your old password incorrectly, verify correct error messages. Step 3: - You should be able to log in with your new password Step 4: - Make sure you cannot enter a null Phone number value or a value of 000-000-0000, verify correct error messages. Place an order with your Phone Number and verify it is correct from POS side. Step 5: - Verify you cannot change your email to an email that already exists, also try not to insert correct values, put different new email and confirm email, verify correct error messages.
Post Feedback about Web Site		1. Go to Web ordering, do not Log In 2. Follow the "Web Site" link at the bottom of the page 3. Fill in the form with a wrong Captcha value 4. Post Feedback 5. Put the correct Captcha value and Post the Feedback again 6. Log In and follow the Link for posting a "Web Site" feedback. 7. Fill in the form and Post	Step 4: - Verify you cannot Post a Feedback with the wrong Captcha value Step 7: - Verify no captcha value is required since you are logged in
Email autocompleter		1) Go to register, signup and account page 2) Type an email and check that after you have typed the '@' sign, an autocompleter with various	The autocompleter should always appear

		domains will appear and can be selected	
Tips in web	Add some values to the “web tip suggestions” pref	1) Go to web ordering, create an order and add check the tip area. 2) Click one of the percentage buttons 3) Go back and forth to the payment and checkout pages. 4) Place the order 5) Repeat the above steps for tips you typed by clicking on the tip textField.	1) It should contain the values from the pref 2) Make sure the amount added is correctly calculated based on your order's price 3) The amount should not change 4) Check the order, it should have the tip
Address autocompleter	Address auto completer should return relevant results. This needs to be tested for USA and International customer. Test with and without prefix N,NE,S,SW... With and without suffix and street type. Streets without ZIPS should not be displayed Test international street format.	There to many possible test cases here. Open web ordering and fill addresses in /account ,/intro and /checkout pages Try some that use the following pattern 1) Input 120 Br 2) Input 91 Main 3) Input 25 Test1 Rd 4) Input 398 S S 5) Test the above when international street form is enabled Also in general some requirements 120 Main - >should return 120 South Main ,120 Main, 120 NE Main 230 S Main -> Should return 120 South Main,120 South East Main. 340 Main Av -> Should return street named Main of type that begins with Av Note that if a street is named south,north the address finder will not work correctly.	1) Should return results like 120 Brandon, Townsend, De, 19734, 120 S Broad St, Middletown, De, 19709 ,120 N Broad St, Middletown, De, 19709,120 Brooks Ct, Middletown, De, 19709 2) Should return results like 91 Main St, Middletown, De, 19709 , 91 W Main St, Middletown, De, 19709 , 91 Main St, Warwick, Md, 21912 3) Should return results like 25 Test1 Rd, Middletown, De, 19709 , 25 Test1 Rd, San Antonio, Pr, 00690 4)Should return results like 398 S Saint Augustine Rd, Warwick, Md, 21912 , 398 S Scott St, Middletown, De, 19709. 5) If you give the number in the same box with the address no results should return. The number should be

			filed in his field for the address finder to return the results.

ΕΠΙΛΟΓΟΣ

Τα test cases είναι ένα κομμάτι που συνεχώς θα εξελίσεται και θα τροποποιείται. Αν σταματήσει η συντήρησή τους, τα test cases μπορεί να πάψουν να είναι σωστά και αποδοτικά. Ειδικά για ένα σύστημα που αναπτύσσεται τόσο γρήγορα είναι σημαντικό να κρατούνται και τα test cases updated.

Για κάθε νέα αλλαγή του κώδικα του συστήματος πρέπει να δημιουργείται και το αντίστοιχο test case έτσι ώστε να υπάρχει μία σύνδεση και ο έλεγχος που θα γίνεται να είναι σωστός και να ανταποκρίνεται στην πραγματικότητα.

ΚΕΦΑΛΑΙΟ 4

Παρουσίαση του κώδικα

ΕΙΣΑΓΩΓΗ

Για να μπορέσουμε να σχεδιάσουμε τον κώδικα χρειάζεται να διαχωρίσουμε τις κλάσεις σε αυτές που θα μας προσφέρουν την διαχείριση, τον χειρισμό και την αναζήτηση των στοιχείων και σε αυτές που θα περιέχουν τον test κώδικα.

Για το κομμάτι των κλάσεων που θα περιέχουν τον κώδικα του Selenium θα ακολουθήσουμε το Page Object model. Θα αντιμετωπίζουμε κάθε ξεχωριστή σελίδα σαν ένα ξεχωριστό αντικείμενο το οποίο θα μας προσφέρει με μεθόδους πρόσβαση στα μέρη του, τα οποία μπορεί να είναι φόρμες, input fields, κ.α.

Με την ύπαρξη των απαραίτητων εργαλείων και μεθόδων για την σύνταξη των test cases θα είναι πολύ γρήγορο και εύκολο για τον QA Tester να δημιουργήσει και να προσθέσει νέα test cases. Σκοπός μας είναι η δημιουργία ενός set από μεθόδους οπού μπορούν να δώσουν αυτή την ευκολία στον κάθε tester να δημιουργήσει έυκολα και απλά τα test cases που χρειάζεται.

Θα χωρίσουμε τον κώδικα μας σε τρία πακέτα. Ένα πακέτο για τις κλάσεις που θα τις χρησιμοποιήσουμε για το setup, ένα πακέτο για τις κλάσεις των εργαλείων του navigation, και ένα πακέτο για τα tests τα οποία θα εξελίσσονται συνεχώς. Για το τελευταίο θα παρουσιάσουμε κάποια παραδείγματα

Η σύνταξη του κώδικα έχει γίνει στο Eclipse IDE με πρόσθετα το Selenium 2.0 και JUnit 4.

ΥΠΟΚΕΦΑΛΑΙΟ 4.1

Setup Classes

Αυτές οι κλάσεις θα μας βοηθήσουν για να χειριστούμε κυρίως τα preferences του συστήματος και να ορίζουμε κάθε φορά το setup που χρειαζόμαστε για να τρέξουμε τα tests μας.

Settings.java

```
package setup;

public class Settings {

    //------------------ Test Suite -------------------
    public static boolean TAKEFAILURESCREENSHOTS = false;

    //------------------ Web Ordering -------------------
    public static String LABURL = "https://localhost:9443";
    public static String SUPERUSERNAME = "Tech Support 75";
    public static String SUPERPASSWORD = "jordan123";
    public static String SUPEREMAIL = "jordan.bozas+1@foodtecsolutions.com";


    //------------------ Web Reports -------------------

    //Only for initializing static variables
    public Settings(){}

}
```

Η συγκεκριμένη κλάση περιέχει configuration στοιχεία τα οποία χρειαζόμαστε για να τρέξουμε τα test μας.

Το LABURL μας δίνει το url στο οποίο θα τρέξουμε τα test μας.

Το SUPERUSERNAME, SUPERPASSWORD, SUPEREMAIL είναι τα credentials για να κάνουμε login σαν administrators στο σύστημα και να μπορέσουμε να κάνουμε edit τα preferences.

Μπορούμε εύκολα να προσθέσουμε ότι άλλη πληροφορία μπορεί να χρειαστούμε στο configuration file μας εύκολα.

Prefs.java

```
package setup;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.Select;

public class Prefs {

    private WebDriver driver;
    private String name;
    private String value;
    private String group;


    public Prefs() {}

    public Prefs(WebDriver driver, String name, String group, String value) {
        this.driver = driver;
        this.name = name;
        this.group = group;
        this.value = value;
    }

    public void update() {


        this.driver.get(Settings.LABURL + "/reports/");
        driver.findElement(By.linkText("Prefs Editor")).click();
        new Select(driver.findElement(By.name("group"))).selectByVisibleText(this.group);
        driver.findElement(By.id("filterTerms")).clear();
        driver.findElement(By.id("filterTerms")).sendKeys(this.name);
        driver.findElement(By.name("edit")).click();



        /*
         * driver.findElement(By.xpath("(//input[@name='value'])[2]")).click();
```

```
        * driver.findElement(By.cssSelector("form.inline > input[type=\"submit\"]")).click();
        * driver.findElement(By.name("value")).click();
        */

    //Save Changes
    driver.findElement(By.cssSelector("form.inline > input[type=\"submit\"]")).click();
}
```

```
public void chooseOption(WebElement list, String value) {
        Select options = new Select(list);
        options.selectByVisibleText(value);
}
```

Αυτή η κλάση μας βοηθάει να ορίσουμε την τιμή που θέλουμε για το preference του συστήματος που θέλουμε να αλλάξουμε. Ο δομητής της κλάσης δέχεται τα properties driver όπου είναι ο webdriver του Selenium που θα χρησιμοποιήσουμε, name όπου είναι το όνομα του preference που θέλουμε να κάνουμε edit, group όπου είναι η ομάδα των preferences που ανήκει στο preference που θέλουμε να κάνουμε edit, και value που είναι η τιμή του preference που θέλουμε να ορίσουμε.

Η μέθοδος update() χρησιμοποιώντας εντολές του Selenium Web Driver κατευθύνεται στον Pref Editor του συστήματος, αναζητάει το preference με το όνομα του και τέλος κάνει update το value που του έχουμε ορίσει.

ΥΠΟΚΕΦΑΛΑΙΟ 4.2

Navigation Classes

Όπως αναφέραμε και στην εισαγωγή η δομή που θα ακολουθήσουμε για να σχεδιάσουμε τις Navigation Classes μας είναι αυτή του Page Object Model. Κάθε βασική σελίδα του συστήματός μας θα αντιπροσωπεύεται από ένα αντικείμενο το οποίο με τις μεθόδους που θα δημιουργήσουμε θα μας προσφέρει πρόσβαση σε όλα τα στοιχεία του.

MenuPage.java

```
package navigation.WebOrdering;


import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import setup.Settings;


public class MenuPage {

    public static boolean usePopUpSizeSelector = true;

    public static void clickMenu(WebDriver driver){
        driver.get(Settings.LABURL+"/menu");
    }

    public static void clickMyOrders(WebDriver driver){
        driver.get(Settings.LABURL+"/orders");
    }

    public static void clickLoyaltyTab(WebDriver driver){
        driver.findElement(By.className("loyalty-tab aTab ")).click();

   }

    public static void clickMyAccount(WebDriver driver){
        driver.get(Settings.LABURL+"/account");
    }
```

```
    public static void clickHelp(WebDriver driver){
        driver.get(Settings.LABURL+"/help");
    }

    public static void clickCategory(WebDriver driver,String CategoryName){
       driver.findElement(By.id(CategoryName)).click();
    }

    public static void navigateCategory(WebDriver driver,String CategoryName){
       driver.get(Settings.LABURL + "/menu?category="+CategoryName);
    }

    public static void clickApplyCouponButton(WebDriver driver){
        driver.findElement(By.id("applyCouponLink")).click();
    }

    public static void clickCheckout(WebDriver driver){
        driver.findElement(By.id("checkOrderLink")).click();
    }

    public static void AddItemByPrice(String category, String price, WebDriver driver,
String baseUrl){
                //Navigate to the Menu Category
                driver.get(baseUrl + "/menu?category="+category);

                //Click on the Price link of the item
                driver.findElement(By.linkText(price)).click();

                //Press Done button
                driver.findElement(By.xpath("//button[@alt=\"done editing\"]")).click();



        }

        public static void AddItemByName(String category, String name, String size,
WebDriver driver, String baseUrl){

                //Navigate to the Menu Category
                driver.get(baseUrl + "/menu?category="+category);

                //Click on the items Name
                driver.findElement(By.xpath("//div[@class=\"itemLabel\" and
contains(text(),'"+name+"')]")).click();

                //Click on the items Size button
                driver.findElement(By.xpath("//div[@class=\"sizesDropDownButton\" and
contains(text(),'"+size+"')]")).click();
```

```
                //Press Done button
                driver.findElement(By.xpath("//button[@alt=\"done editing\"]")).click();


        }

    public static void removeOrderedItem(WebDriver driver,String itemName){
        driver.findElement(By.partialLinkText("javascript:removeItem(*"+itemName));
    }

}
```

Η κεντρική σελίδα του Web Ordering είναι η MenuPage. Αυτή είναι η αφετηρία για κάθε action στη σελίδα μας. Έχουμε τις μεθόδους με τις οποίες μπορούμε να περιηγηθούμε στις υπόλοιπες σελίδες.

Αρχικά έχουμε τις μεθόδους με τις οποίες μπορούμε να χειριστούμε το navigation menu και να κατευθυνθούμε στις σελίδες: Menu, My Orders, Loyalty, My Account και Help.

Με κάθε μέθοδο μπορούμε να κατευθυνθούμε σε αυτές τις σελίδες.

Επίσης έχουμε μεθόδους για να χειριστούμε το δεξί column της σελίδας και να επιλέξουμε τα button actions.

Τέλος έχουμε την δυνατότητα να επιλέξουμε MenuCategory και να προσθέσουμε MenuItem στην παραγγελία μας.

Login.java

```
package navigation.WebOrdering;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;


public class Login {

        String url ="/login";
        String reports = "/reports";

        public String getLoginUrl(){
                return this.url;
        }

        public String getReportsUrl(){
                return this.reports;
```

```
        }

    public static void loginWebOrdering(WebDriver driver,String email,String password){

        driver.findElement(By.id("Email")).clear();
       driver.findElement(By.id("Email")).sendKeys(email);
       driver.findElement(By.id("Password")).clear();
       driver.findElement(By.id("Password")).sendKeys(password);
       driver.findElement(By.xpath("//button[@id='login-btn']")).click();

    }


   public static void facebookLogin(WebDriver driver,String email,String password){

        driver.findElement(By.id("facebook-login-btn")).click();
    }


    public void loginReports(WebDriver driver, String supportId, String password) {
        driver.findElement(By.id("LoginName")).clear();
        driver.findElement(By.id("LoginName")).sendKeys(supportId);
        driver.findElement(By.id("LoginPasswd")).clear();
        driver.findElement(By.id("LoginPasswd")).sendKeys(password);
        driver.findElement(By.id("LoginButton")).click();
    }

    public static boolean checkIfAuthed(WebDriver driver, String supportId) {
        try{
            boolean loggedIn = driver.getPageSource().contains("Logged in as
"+supportId);
            return loggedIn;
          }
          catch(Exception e){
            return false;
          }

    }

}
```

Η συγκεκριμένη σελίδα μας επιτρέπει να κάνουμε login είτε στο Web Ordering σαν πελάτες, είτε στο Reports panel όπου μπορούμε να κάνουμε login σαν Administrators. Επίσης έχουμε την δυνατότητα να κάνουμε login με το Facebook account μας.

Σε κάθε περίπτωση έχουμε στην διάθεσή μας μεθόδους που μας βοηθούν να κάνουμε login στο σύστημα περνώντας σαν παραμέτρους τα στοιχεία μας.

Τέλος έχουμε την μέθοδο checkIfAuthed() η οποία μας επιστρέφει αν ένας χρήστης είναι logged in.

EditItemPage.java

```
package navigation.WebOrdering;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;


public class EditItemPage {


    public static void clickItem(WebDriver driver, String itemName){
        driver.findElement(By.linkText(itemName)).click();
    }

    public static void clickIngredient(WebDriver driver, String ingredientName){
        driver.findElement(By.id(ingredientName)).click();
    }

    public static void clickQualifier(WebDriver driver, String ingredientName){

driver.findElement(By.xpath("//span[contains(text(),'"+ingredientName+"')]/following-sibling::span")).click();
    }

    public static void addQualifier(WebDriver driver, String qualifierName){
        driver.findElement(By.xpath("//div[@class='qualChoice' and contains(text(),'"+qualifierName+"') and not(contains(@style,'display:none'))]")).click();
    }

    public static void expandAccordion(WebDriver driver,String categoryName){
        driver.findElement(By.linkText(categoryName)).click();
    }

    public static void doneEdit(WebDriver driver){
        driver.findElement(By.xpath("//button[@onclick='location.href=\"/\"']")).click();
    }
```

```
    public static boolean isIngredientChoiceMissing(WebDriver driver){
        if(driver.findElement(By.xpath("//div[@id='order-errors']/div")) != null){
            return true;
        }
        return false;
    }


}
```

Η συγκεκριμένη σελίδα μας επιτρέπει να διαμορφώσουμε τα items που έχουμε προσθέσει στην παραγγελία μας, να προσθέσουμε ingredients καθώς και να αφαιρέσουμε.

CreateNewAddress in AccountPage.java

```
public static void createNewAddress(WebDriver driver, String type, String location,
String address, String diff, String notes) throws InterruptedException{
            driver.get(Settings.LABURL + accountPageURL);
        driver.findElement(By.linkText("Create a new address")).click();
        for (int second = 0;; second++) {
            if (second >= 60) fail("timeout");
            try { if (isElementPresent(By.id("ui-dialog-title-AddressOptionsDialog")))
break; } catch (Exception e) {}
            Thread.sleep(1000);
        }

        for (int second = 0;; second++) {
            if (second >= 60) fail("timeout");
            try { if (isElementPresent(By.id("LocationName"))) break; } catch
(Exception e) {}
            Thread.sleep(1000);
        }

        driver.findElement(By.name("LocationName")).clear();
        driver.findElement(By.name("LocationName")).sendKeys(location);
        driver.findElement(By.cssSelector("#AddressTypeField >
#AddressType")).click();
        new Select(driver.findElement(By.cssSelector("#AddressTypeField >
#AddressType"))).selectByVisibleText(type);
        driver.findElement(By.cssSelector("option[value=\""+type+"\"]")).click();
        driver.findElement(By.id("Street")).click();
        driver.findElement(By.id("Street")).sendKeys(address);
        for (int second = 0;; second++) {
            if (second >= 60) fail("timeout");
            try { if (isElementPresent(By.id("street_autocomplete"))) break; } catch
```

```
(Exception e) {}
                Thread.sleep(1000);
            }

            driver.findElement(By.xpath("//div[@id='street-list-container']/ul/li")).click();
            driver.findElement(By.id("Value1")).click();
            driver.findElement(By.id("Value1")).clear();
            driver.findElement(By.id("Value1")).sendKeys(diff);
            driver.findElement(By.id("Instructions")).click();
            driver.findElement(By.id("Instructions")).clear();
            driver.findElement(By.id("Instructions")).sendKeys(notes);
            driver.findElement(By.xpath("(//button[@type='button'])[2]")).click();

        }
```

Από τα πιό σημαντικά μέρη της AccountPage είναι η διαχείριση των αποθηκευμένων διευθύνσεων. Με την μέθοδο CreateNewAddress μπορούμε να δημιουργήσουμε μία νέα διεύθυνση και να την αποθηκεύσουμε στον λογαριασμό του χρήστη.

Μπορούμε να χρησιμοποιήσουμε το autocomplete feature για να προσθέσουμε μία διεύθυνση στον λογαριασμό μας. Από εκεί και πέρα συμπληρώνουμε τα πεδία της φόρμας για να αποθηκεύσουμε την διεύθυνση.

ΥΠΟΚΕΦΑΛΑΙΟ 4.3

Test Classes

Για κάθε test case θα μπορούμε να έχουμε μία ξεχωριστή test κλάση. Μπορούμε να τις ομαδοποιήσουμε όπως θέλουμε σε σουίτες αναλόγως με τα test που θέλουμε να τρέξουμε.

Παρακάτω ακολουθούν κάποια παραδείγματα με την δομή και την χρήση των navigation κλάσεων.

Μπορούμε να χρησιμοποιήσουμε μία BasicTest κλάση ως βάση και στη συνέχεια να την κληρονομεί κάθε test μας έτσι ώστε να έχουμε μία βασική δομή.

BasicTest.java

```
package selenium.Tests;

import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
```

```
import org.openqa.selenium.support.ui.Select;

import setup.Settings;

public class BasicTest {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = Settings.LABURL;
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test


  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
```

```
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

Η βασική δομή μας επιτρέπει να χρησιμοποιήσουμε την configuration setup κλάση για να έχουμε τις πληροφορίες που χρειαζόμαστε για να τρέξουν τα τεστ μας.

Η @Before setUp() μέθοδος ορίζει τις παραμέτρους που είναι συγκεκριμένες καθώς και την αρχικοποίηση του WebDriver που θα χρησιμοποιηθεί.

Σε αυτό το σημείο μπορούμε να ορίσουμε και τα preferences, αλλά αυτό πρέπει να γίνει για κάθε test ξεχωριστά.

Στη συνέχεια ακολουθούν όλα τα τεστ με το annotation @Test.

Τέλος έχουμε κάτω από το annotation @After την μέθοδο tearDown() όπου επαναφέρουμε το setup μας στην αρχική του μορφή.

Παράλληλα έχουμε δημιουργήσει κάποιες γενικά χρήσιμες μεθόδους οι οποίες μπορούν να μας προσφέρουν ευκολία για τον έλεγχο κάποιων συνθηκών. Αυτές είναι οι isElementPresent(), isAlertPresent, closeAlertAndGetItsText().

UnregisteredUser.java

```
package selenium.Tests;

import static org.junit.Assert.*;

import java.util.concurrent.TimeUnit;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import navigation.WebOrdering.Login;
```

```
import setup.Prefs;

public class UnregisteredUser {

	private WebDriver driver;
	private String baseUrl;
	private boolean acceptNextAlert = true;
	private StringBuffer verificationErrors = new StringBuffer();


	@Before
	public void setUp() throws Exception {
	   driver = new FirefoxDriver();
	   baseUrl = "https://localhost:9443/";
	   driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
	   if (!Login.checkIfAuthed(driver, "Tech Support 75")){
	        Login authed = new Login();
	        authed.loginReports(driver, "Tech Support 75", "jordan123");
	   }

	}

	@After
	public void tearDown() throws Exception {
	   driver.quit();
	   String verificationErrorString = verificationErrors.toString();
	   if (!"".equals(verificationErrorString)) {
	     fail(verificationErrorString);
	   }
	}

	@Test
	public void test() {
	        Prefs allowUnregistered = new Prefs(driver, "Allow unregistered users",
"Web Order", "True");
	        allowUnregistered.update();


	}

}
```

ΕΠΙΛΟΓΟΣ

Έχοντας την δυνατότητα να χειριστούμε κάθε στοιχείο της σελίδας μπορούμε εύκολα και γρήγορα να δομήσουμε ένα test ώστε να ελέγξουμε την λειτουργικότητα του συστήματός μας.

Αυτό μας δίνει την δυνατότητα να έχουμε ένα μεγάλο πλήθος από test cases τα οποία μπορούν να τρέξουν εύκολα και γρήγορα μαζικά και να προσφέρουν μία πλήρη εικόνα για την λειτουργικότητα του συστήματος.

Σύγκριση των δύο διαδικασιών

Εισαγωγή

Η χειροκίνητη και η αυτοματοποιημένη δοκιμή και οι δύο έχουν τη δική τους αξία και σημασία . Όπως αναφέρθηκε το χειροκίνητο και αυτόματο τεστινγκ και τα δύο έχουν τις δικές τους αξίες , συνεπώς, θα συζητήσουμε αυτές τις αξίες και τη σημασία στον κόσμο του λογισμικού .

Manual Testing

Όπως γνωρίζουμε το χειροκίνητο τεστ είναι ένα από τα παλαιότερα και πιο συχνά χρησιμοποιούμενες τεχνικές στις δοκιμές του λογισμικού . Ακόμη και τώρα , ενώ εκτελείτε η αυτοματοποιημένη δοκιμή θα πρέπει να ελεγχθεί με μη αυτόματο τρόπο η εφαρμογή , τουλάχιστον μία φορά , ως εκ τούτου, με άλλα λόγια μπορούμε να πούμε ότι η αυτοματοποιημένη δοκιμή δεν είναι πλήρης χωρίς χειροκίνητη δοκιμή. Οι δύο τεχνικές λειτουργούν ταυτόχρονα . Η αυτοματοποιημένη δοκιμή μπορεί να αποτύχει , αν ένας απλό και απρόσμενο pop- up παράθυρο εμφανίστει. Αυτή η περίπτωση μπορεί να αναλυθεί γρήγορα και να απορριφθεί , εάν η δοκιμή γίνεται με το χέρι . Το χειροκίνητο τέστινγκ είναι μια δραστηριότητα που πραγματοποιείται με βάση την ανθρώπινη κρίση και ανάλυση . Στη χειροκίνητη δοκιμή πρέπει ο tester να κατέχει καλή γνώση του συστήματος και να κρίνει αν η εφαρμογή υπό δοκιμή λειτουργεί όπως αναμενόταν ή όχι . Με χρήση χειροκίνητης δοκιμής ένα μεγάλο ποσό από τα σφάλματα μπορούν να αναγνωριστούν σε σχέση με την αυτοματοποιημένη δοκιμή . Με την χρήση του manual testing μπορείτε να εξερευνήσετε και να βρείτε τους τομείς που μπορεί να μην έχουν δοκιμαστεί.

α) Το manual testing μπορεί να βρεί layout προβλήματα .

β)Σύνθετα σφάλματα μπορούν να βρεθούν μόνο με manual testing .

γ) Bugs που σχετίζονται με τα θέματα ευχρηστίας μπορούν να προσδιοριστούν μόνο με χειροκίνητο έλεγχο .

δ)Το manual testing βοηθά δοκιμαστές να προσδιορίσουν τη λειτουργικότητα της εφαρμογής τους καλύτερα .

ε) Το manual testing μπορεί να εφαρμοστεί και σε μικρά και σε μεγάλα πρότζεκτ .

στ) Το manual testing είναι πολύ πιο οικονομική και εύκολη διαδικασία.

Το manual testing απαιτεί εμπειρία του συστήματος και γνώση των λειτουργιών του. Ο tester δημιουργεί test cases με βάση την αναμενόμενη λειτουργία του συστήματος.

Ακολουθείται ένα σενάριο επιτυχίας όπου επιστρέφει τα αποτελέσματα που θέλουμε, καθώς επίσης περιγράφονται και οι περιπτώσεις οι οποίες θέλουμε να καλυφθούν ως μή αποδεκτές ενέργειες. Το σύστημα πρέπει να αντιμετωπίζει σωστά περιπτώσεις λάθος ή κακόβουλης εισόδου και να επιστρέφει τα κατάλληλα αποτελέσματα. Όλες αυτές οι περιπτώσεις περιγράφονται στο τεστ.

Automated Testing

Η αυτοματοποιημένη δοκιμή ενισχύει τις πιθανότητες της παραγωγής σφαλμάτων . Ένα από τα πιο καταπληκτικά χαρακτηριστικά της αυτοματοποιημένης δοκιμής είναι ότι μπορεί να εκτελεστεί πολλές φορές χωρίς καμία διακοπή κάτι το οποίο σίγουρα δεν είναι δυνατό με τη χρήση χειροκίνητων δοκιμών . Test cases που έχουν προετοιμαστεί με την χρήση automated testing μπορούν να τρέξουν πιο γρήγορα σε σύγκριση με το manual testing, που σημαίνει ότι τώρα μπορείτε να εκτελέσετε περισσότερες περιπτώσεις δοκιμών σε περισσότερα προγράμματα περιήγησης , για παράδειγμα 10 browsers και 5 συσκευές μπορεί να τρέξoyn γρήγορα και χωρίς κανένα χρονικό κενό, κάτι το οποίο δεν είναι δυνατόν με χειροκίνητο έλεγχο .

Ως εκ τούτου, τα οφέλη των αυτοματοποιημένων δοκιμών που αναφέρονται παρακάτω:

α) Η αυτοματοποιημένη δοκιμή εξοικονομεί χρόνο και ανθρώπινη προσπάθεια .

β) Επιτρέπει δοκιμαστές να εκτελέσουν περισσότερες περιπτώσεις δοκιμής κατ 'επανάληψη χωρίς καμία χρονική υστέρηση και διακοπή .

γ) Αυτοματοποιημένεςδοκιμές βοηθούν δοκιμαστές στην προετοιμασία σε περιπτώσεις δοκιμών που είναι με το χέρι αδύνατο να τρέξουν .

δ) Χρησιμοποιώντας αυτοματοποιημένη δοκιμή , τα ανθρώπινα λάθη μπορεί να αφαιρεθούν εντελώς από τη διαδικασία της δοκιμής.

ε) Η αυτοματοποιημένη δοκιμή βοηθά στην προετοιμασία λεπτομερών περιπτώσεων δοκιμών , ακόμη και σε μη αυτοματοποιημένες περιοχές .

Acceptance Tests

Τα acceptance tests βασίζονται στα user stories. Κατά τη διάρκεια ανάπτυξης μίας νέας έκδοσης τα user stories θα μεταφραστούν σε acceptance tests. Ο πελάτης ορίζει από τις ανάγκες του τα σημεία που θα πρέπει να καλύπτονται από το acceptance test. Ένα user story μπορεί να έχει ένα ή περισσότερα acceptance tests.

Τα acceptance tests είναι tests μαύρου κουτιού (black box testing), δηλαδή ο tester δεν γνωρίζει την λειτουργικότητα του συστήματος όταν τρέχει ένα τεστ. Περιμένει ένα συγκεκριμένο αποτέλεσμα περνώντας μία είσοδο. Κάθε acceptance test

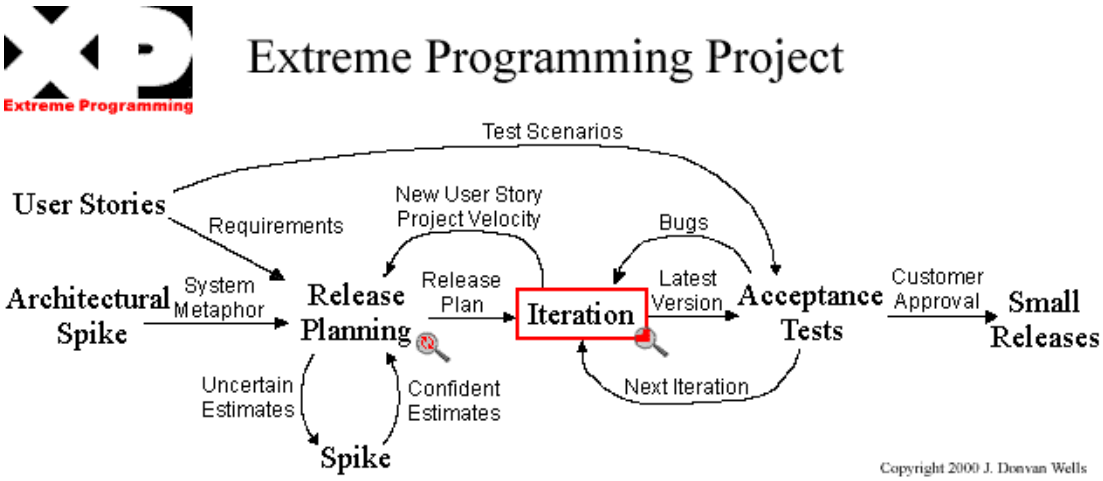
αντιπροσωπεύει ένα συγκεκριμένο αποτέλεσμα από το σύστημα. Τα acceptance tests χρησιμοποιούνται σαν regression tests πριν από κάθε μεγάλο release.

Η διασφάλιση της ποιότητας (QA) είναι ένα ουσιαστικό μέρος της διαδικασίας ανάπτυξης . Σε ορισμένα έργα το QA γίνεται από μια ξεχωριστή ομάδα , ενώ σε άλλα το QA θα είναι ένα μέρος στην ομάδα ανάπτυξης.

Τα acceptance tests πρέπει να είναι αυτοματοποιημένα, έτσι ώστε να μπορούν να τρέξουν συχνά. Είναι ευθύνη της ομάδας να προγραμματίσει το χρόνο σε κάθε επανάληψη για να διορθώσει τυχόν αποτυχημένες δοκιμές.

Τα acceptance tests άλλαξαν όνομα από functional tests. Αυτό αντανακλά καλύτερα τον σκοπό τους, ο οποίος είναι να διασφαλισθεί ότι οι απαιτήσεις των πελατών έχουν εκπληρωθεί και το σύστημα είναι αποδεκτό.

Όπως παρουσιάζεται και στο παρακάτω διάγραμμα που αντιπροσωπεύει την διαδικασία ανάπτυξης ενός XP (Extreme Programming) project τα acceptance tests είναι το τελευταίο σύνορο πριν το σύστημα πάει στον πελάτη, και διασφαλίζει την σωστή λειτουργικότητα του συστήματος.



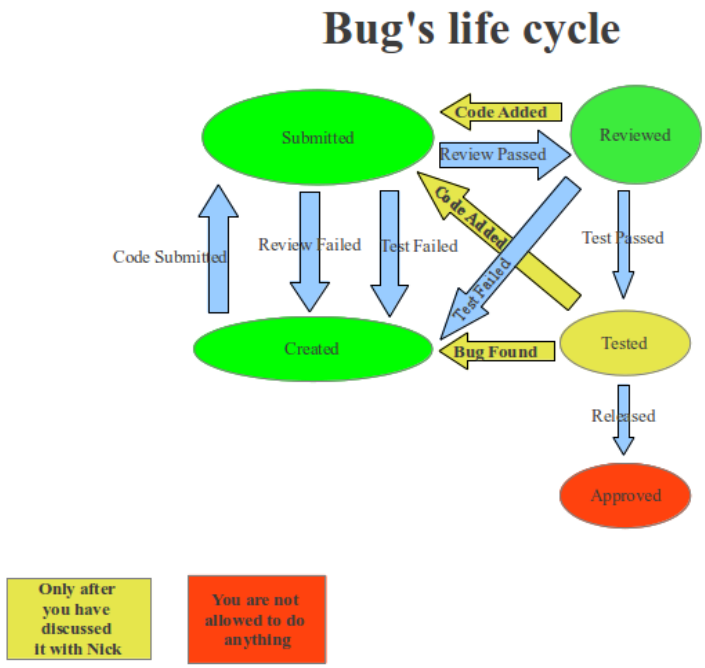
Παρουσίση των στοιχείων της εταιρίας

Η εταιρία πριν από κάθε major release διαθέτει πόρους (ανθρώπινο δυναμικό) και χρόνο έτσι ώστε να τρέξουν κάποια tests για να είναι σίγουρη η τελική λειτουργικότητα του προιόντος. Αυτή η διαδικασία είναι γνωστή και ως regression testing.

Σε αυτό το χρονικό διάστημα η πλειοψηφία του QA department αφιερώνει τον χρόνο της σε αυτή τη διαδικασία. Για να ελεγχθεί το κομμάτι του συστήματος που μελετάμε (Web Ordering) με manual testing χρειάζονται δύο άτομα του QA να απασχοληθούν για 8-12 εργάσιμες ώρες.

Όπως καταλαβένετε αυτή η διαδικασία είναι αρκετά χρονοβόρα κατά την εκτέλεση, μιας και η εκτέλεση γίνει χειροκίνητα. Δεν υπάρχει κάποια αυτοματοποιημένη διαδικασία για την εκτέλεση αυτών των tests.

Πέρα από τον χρόνο εκτέλεσης αν εμφανιστούν λάθη ακολουθείται μία διαδικασία επαλήθευσης του προβλήματος και αν αυτό το πρόβλημα αναγνωριστεί ως σημαντικό τότε χρειάζεται να επιλυθεί στο συντομότερο.



Η παραπάνω εικόνα απεικονίζει την διαδικασία εξέλιξης ενός bug. Το bug έχει πέντε διαφορετικά statuses.

- Created - Το bug έχει μόλις δημιουργηθεί ή έχει γίνει fail και έχει γυρίσει στο αρχικό του status
- Submitted - Ο developer μόλις έχει κάνει submit το bug ή μία διόρθωση

- Reviewed - Ο reviewer έχει κρίνει ότι οι αλλαγές από τον developer ακολουθούν το coding style και έχει χρησιμοποιήσει σωστές πρακτικές
- Tested - Ο tester έχει τρέξει τα acceptance test του manually και έχει κρίνει ότι το σύστημα έχει την σωστή συμπεριφορά
- Approved - Ο project manager κάνει approve το bug και είναι έτοιμο για release

Όλη την υπόλοιπη περίοδο χρειάζεται να σηντιρούνται αυτά τα test cases ούτως ώστε να ανταποκρίνονται στις τελευταίες αλλαγές του κώδικα. Για να γίνεται σωστά αυτή η συντήρηση πρέπει να αφιερώνεται χρόνος από τους testers του QA για να ανανεώνουν τα test cases.

Το web ordering (το κομμάτι που αναλύουμε) είναι μία υποκατηγορία του γενικού προιόντος της εταιρίας. Ακολουθούν κάποια γενικά στοιχεία για το συνολικό προιόν της εταιρίας.

Χρήσιμα στοιχεία

Major Release	8-10 μήνες
Bugs	250-300
Test Cases	150-200
Web Ordering Test Cases in last version	93
QA Department	6 άτομα

Πίνακας 1

Μετά από μία εκτέλεση των tests αυτά είναι τα πραγματικά αποτελέσματα της εκτέλεσης του Web Ordering Regression στις 31/5/13-4/6/13

Number of test cases	Successful	Failed	Blocked
93	74	11	7

Πίνακας 2

Παραπάνω απεικονίζονται τα εξής στοιχεία:

- Number of test cases= Ο αριθμός των test cases που χρησιμοποιήθηκαν κατά το manual testing του τελευταίου regression.
- Successful= Τα επιτυχή test cases
- Failed= Τα αποτυχημένα test cases
- Blocked= Τα test cases τα οποία έχουν αποτύχει αλλά ο λόγος δεν είναι ξεκάθαρος. Μπορεί ο λόγος να είναι λάθος setup ή όντως να υπάρχει πρόβλημα.

Μετά απο review του προβλήματος αποφασίζεται αν το test είναι επιτυχές ή όχι.

Ο κύκλος εκδόσεων του συστήματος (Release Cycle)

Major Releases

Release	Bugs	Created	Released	Bugs for Web Ordering	Enhancements for Web Ordering
9.0.0	374 Bugs	8/20/13	1/27/14	51	37
8.3.0	284 Bugs	2/20/13	6/7/13	34	12

Πίνακας 3

- Release - Ο αριθμός του version του release
- Bugs - Ο αριθμός των bugs του συνολικού προιόντος για αυτό το release
- Created - Η ημερομηνία που ξεκίνησε να γίνεται develop το release
- Released - Η ημερομηνία που έγινε το release και το προιόν πήγε στον πελάτη
- Bug for Web Ordering - Ο αριθμός των bugs που φτιάχτηκαν στο release για το module που αναλύουμε (web ordering)
- Enhancements for Web Ordering - Ο αριθμός των νέων λειτουργιών που προστέθηκαν στο release για το module που αναλύουμε (web ordering)

Minor Releases

Οι μικρές εκδόσεις έχουν έναν κύκλο development στις δύο εβδομάδες.

Δυναμικό της εταιρίας

Developers: 15 developers
QA: 6 in QA

Τελευταία αποτελέσματα από το regression της 9.0.0

Web Ordering (Χρησιμοποιώντας Automated testing)

Test Cases Failed	Test Cases Passed	To Be Reviewed	% Success
9	74	8	81%

Πίνακας 4

Target Market(Χρησιμοποιώντας Manual testing)

Test Cases Failed	Test Cases Passed	To Be Reviewed	% Success
1	22	2	88%

Πίνακας 5

Ο παρακάτω πίνανακας δείχνει τα test cases που έχουν γίνει fail για κάθε module (με κόκκινο) και τα test cases που χρειάζοντια review (με κίτρινο). Παραθέτετε ο αριθμός του test case για γρήγορη εύρεση. Αν ένα test case έχει αποτύχει και δεν είναι ξεκάθαρο το πρόβλημα περνάει από review γιατί μπορεί να είναι false positive.

Module	Testcase
Web	19
Web	24
Web	25
Web	29
Web	30
Web	32
Web	36
Web	38
Web	41
Web	42
Web	47
Web	77
Web	82
Web	87
TargetMarket	20
Web	50

Πίνακας 6

Ύστερα από την εκτέλεση των tests καταλήγουμε στα παρακάτω αποτελέσματα (οι αριθμοί αναφέρονται για το κομμάτι του Web Ordering)

Στοιχεία Web Ordering (Manual and Automated)

	Χρόνος	Επιτυχία	Λάθη	False Positives	Κάληψη	Επαναχρησιμοποίηση
Manual Tests	15 ώρες	88%	9	8	70%	0%
Automated Tests	45 λεπτά	81%	6	10	85%	100%

Πίνακας 7

Στοιχεία Target Market (Only Manual Testing)

	Χρόνος	Επιτυχία	Λάθη	False Positives	Κάληψη	Επαναχρησιμοποίηση
Manual Tests	6 ώρες	88%	1	2	70%	0%

Πίνακας 8

Όπου false positive θεωρείται μία περίπτωση αποτυχίας η οποία δεν είναι πραγματική. Κατά τη διάρκεια εκτέλεσης ενός τεστ μπορεί να προκληθεί ένα σφάλμα το οποίο να προκαλέσει την αποτυχία του τεστ, αλλά να μην έχει να κάνει με την λειτουργικότητα του συστήματος. Επομένως είναι μία αποτυχία η οποία μπορεί να αγνοηθεί ύστερα από έλεγχο και το τεστ να θεωρηθεί ως επιτυχές

Το ποσοστό επιτυχίας υπολογίζεται με βάση τα επιτυχή τεστ έναντι των λανθασμένων. Αν υποθέσουμε ότι έχουμε 100 τεστ που θα τρέξουν και τα 70 είναι επιτυχή, έχουμε ένα ποσοστό επιτυχίας 70% με 30% αποτυχία.

Κάθε σφάλμα κατά την εκτέλεση πρέπει να περάσει από έλεγχο για να επιβεβαιωθεί αν όντως είναι πρόβλημα.

Σύγκριση των δύο μεθόδων

Συγκρίνοντας τα δύο αυτά modules μπορούμε να συμπεράνουμε τα παρακάτω για την κάθε μέθοδο testing:

- Ο χρόνος για την εκτέλεση των automated tests είναι ξεκάθαρα πολύ λιγότερος
- Το manual testing μπορεί να ανακαλύψει κρυμμένα bugs γιατί βοηθάει η εμπειρία και η κριτική ικανότητα του tester, αλλά τα automated tests προσφέρουν επίσης αξιόπιστα αποτελέσματα.

- Τα automated tests μπορούν να προκαλέσουν περισσότερα false positives. Η παραμικρή αλλαγή στην ροή του τεστ μπορεί να κάνει το τεστ να αποτύχει.
- Έχουμε καλύτερη κάληψη του περιεχομένου του προιόντος με τα automated tests. Είναι όλα διατυπομένα και δεν υπάρχει η περίπτωση να ξεχαστεί και να μην τρέξει ένα τεστ σε αντίθεση με το manual testing.
- Τα automated tests μπορούν να επαναχρησιμοποιηθούν ανά πάσα στιγμή εύκολα και γρήγορα. Τα manual tests απαιτούν τον ίδιο χρόνο και κόπο κάθε φορά.

Διαδικασία ανάπτυξης με χειροκίνητα Tests

1. Γραφή κώδικα
2. Upload του κώδικα σε κάποιο depot
3. Build
4. Χειροκίνητη εκτέλεση του κώδικα και των τέστ (πολλές φορές γεμίζωντας φόρμες βήμα-βήμα)
5. Έλεγχος στα Log files, βάση δεδομένων, εξωτερικές υπηρεσίες, τιμές των μεταβλητών, output στην οθόνη κλπ.
6. Αν δεν λειτουργεί σωστά, επανάληψη της προηγούμενης διαδικασίας

Διαδικασία ανάπτυξης με αυτοματοποιημένα Unit Tests

1. Γραφή ενός ή περισσοτέρων test cases
2. Auto-compile και run για να φανεί ότι τα tests κάνουν fail
3. Γραφή κώδικα για να περάσουν τα tests
4. Auto-compile και run
5. Αν τα tests γίνουν fail -> κάνετε τις απαραίτητες αλλαγές
6. Αν τα tests περάσουν -> επανάληψη για την επόμενη μέθοδο

Διαδικασία ανάπτυξης με αυτοματοποιημένα Functional Tests

1. Τελειώνετε με την γραφή κώδικα (με ταl unit tests να έχουν περάσει)
2. Γραφή ενός Functional Test με οποιοδήποτε εργαλείοl
3. Auto-compile και run
4. Αν τα tests γίνουν fail -> κάνετε τις απαραίτητες αλλαγές
5. Αν τα tests περάσουν -> συνέχεια

Προσπάθεια και κόστος

Ας υποθέσουμε ότι έχουμε 6 test cases.

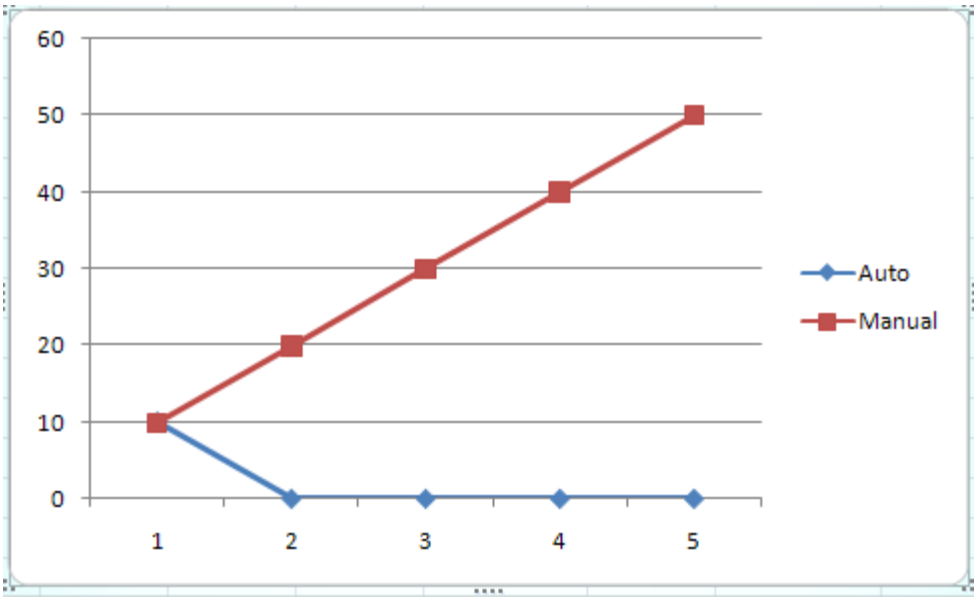
- Η προσπάθεια που χρειάζεται να τρέξουν και τα 6 χειροκίνητα = 30 λεπτά, μέσος όρος 5 λεπτά το κάθε τεστ χειροκίνητα
- Η προσπάθεια που χρειάζεται για να γραφούν acceptance tests και για τα 6 cases = 30 λεπτά, μέσος όρος 5 λεπτά το κάθε τεστ χειροκίνητα
- Η προσπάθεια που χρειάζεται να τρέξουν και τα 6 acceptance tests = λιγότερο από 1 λεπτό
- Επαναλήψεις των τεστ που θα τρέξουν = 5 επαναλήψεις
- Συνολικός χρόνος για να τρέξουν χειροκίνητα = 5*30 λεπτά = 150 λεπτά
- Συνολικός χρόνος για να τρέξουν αυτόματα = 5*1 λεπτό = 5 λεπτά

Ο παρακάτω πίνακας παρουσιάζει το κόστος της χρήσης των manual tests. Παρουσιάζεται για κάθε έκδοση πόσα manual tests χρειάζεται να γραφούν και να τρέξουν, ενώ με τα automated tests δεν χρειάζεται να αφιερωθεί χρόνος για τα test απλά να τρέξουν αυτόματα. Έτσι σε βάθος χρόνου έχουμε σπαταλήσει 5 φορές περισσότερο χρόνο για να τρέξουμε τα manual tests σε σχέση με τα automated tests.

Στον παρακάτω πίνακα απεικονίζεται ο χρόνος που απαιτείται για να τρέξουν 5 επαναλήψεις για 6 τεστ

Release Επαυξήσεις	Manual Test Χρόνος Εκτέλεσης	Auto Test Χρόνος Εκτέλεσης
1 επανάληψη	30 λεπτά	30 λεπτά για συγγραφή + 5 λεπτά για εκτέλεση
2 επανάληψη	30 λεπτά	5 λεπτά
3 επανάληψη	30 λεπτά	5 λεπτά
4 επανάληψη	30 λεπτά	5 λεπτά
5 επανάληψη	30 λεπτά	5 λεπτά
Συνολικός χρόνος	150 λεπτά	55 λεπτά

Πίνακας 9



Γράφημα 1

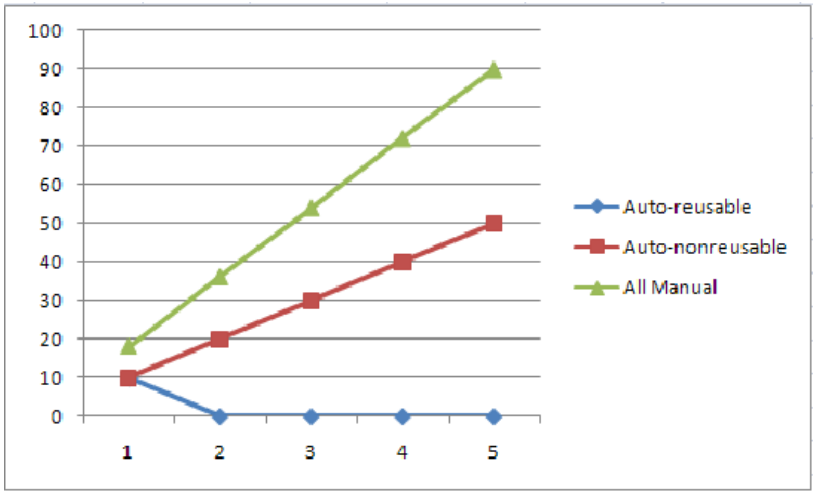
Άξονας Χ: Εξέλιξη εκδόσεων του συστήματος
Άξονας Υ: Συσσώρευση χρόνου. Πόσος χρόνος έχει αφιερωθεί συνολικά για τα test cases

Συμπεράσματα

1. Τα χειροκίνητα tests απαιτούν περισσότερο χρόνο και κόστος σε σχέση με τα Automated Test για να γραφούν και να τρέξουν
2. Το χειροκίνητο Testing είναι βαρετό
3. Τα Automated Tests είναι επαναχρησιμοποιήσημα
4. Τα χειροκίνηταl Tests προσφέρουν μειωμένη εικόνα και χρειάζεται να επαναληφθεί από όλους τους εμπλεκομένους
5. Τα Automated Tests μπορεί να έχουν διαφορετικά πεδία εφαρμογής και να ελέγχουν τις μεμονωμένες μονάδες κώδικα αναλόγως των εξαρτήσεων
6. Τα Automated tests μπορεί να χρειάζονται λιγότερο περίπλοκο setup και teardown
7. Το Automated Testing βεβαιώνει ότι δεν θα ξεχαστεί κάποιο test case
8. Το Automated Testing οδηγεί σε clean design του κώδικα
9. Τα Automated Tests προσδίδουν προστασία για refactoring
10. Τα Automated Tests έχουν αξία ως προς τη μάθηση του συστήματος. Μπορούν να υπάρξουν και ως documentation
11. Τα Automated Tests δεν δημιουργούν ακαταστασία στον κώδικα / κονσόλα / logs

Καλές πρακτικές

- Οργάνωση των test σωστά
- Αν δεν γίνεται commit των tests δεν μπορούν να επαναχρησιμοποιηθούν και το πλεονέκτημα της μειωμένης προσπάθειας χάνεται



Γράφημα 2

Άξονας Χ: Εξέλιξη εκδόσεων του συστήματος
Άξονας Υ: Συσσώρευση χρόνου. Πόσος χρόνος έχει αφιερωθεί συνολικά για τα test cases

Όπως παρουσιάζεται στο γράφημα, σε βάθος χρόνου τα automated tests δεν απαιτούν νέο χρόνο για να εκτελεστούν παρά μόνο τον χρόνο που απαιτούν για την γραφή και την δημιουργία τους. Σε αντίθεση με τα manual tests τα οποία απαιτούν τον ίδιο χρόνο κάθε φορά για τα ξανατρέξει ο tester. Βέβαια βλέπουμε ότι αν τα automated tests γίνουν deprecated και δεν ακολουθούν την λογική του συστήματος χάνουμε σε χρόνο γιατί απαιτείται εκ νέου χρόνος για να τα κάνουμε update.

Επίλογος

Με την αυτοματοποίηση αυτών των διαδικασιών αυτή η διαδικασία μπορεί να διαρκέσει ελάχιστα λεπτά σε σχέση με τις μέρες που απαιτούσε προηγουμένως. Είναι πολύ εύκολο και απλό να τρέξουμε αυτά τα test ανά πάσα στιγμή και όχι μόνο πριν από ένα major release.

Απαιτείται μόνο ένα άτομο για να τρέξει την αυτόματη διαδικασία.
Η αναγνώριση των λαθών είναι πιό εύκολη. Σε αυτό το μέρος όμως χρειάζεται η ίδια διαδικασία για να κριθεί αν ένα λάθος είναι πρόβλημα ή όχι. Αθροιστικά έχουμε βέβαια βελτίωση γιατί μειώνεται ο χρόνος ελέγχου-επιβεβαίωσης-καραγραφής και διόρθωσης. Ίσως μελλοντικά να προστεθεί ένα επιπλέον επίπεδο αυτοματοποίησης για την αναγνώριση και ταυτοποίηση των λαθών.

Η συντήρηση είναι ένα μέρος στο οποίο χρειάζεται περισσότερος χρόνος και ώρα εργασίας. Η απλή καταγραφή ενός test case με τον παλιό τρόπο είναι πιό απλή. Για να δημιουργηθεί ένα αυτοματοποιημένο test case χρειάζεται μία εξοικίωση με το σύστημα και περισσότερος χρόνος.

Σε γενικές γραμμές υπάχει ξεκάθαρη βελτίωση στην απόδοση της διαδικασίας.

Υπάρχουν δυνατότητες για περαιτέρω βελτίωση υλοποιώντας δυνατότητα για cross-browser test compatibility. Αυτή η λειτουργία θα βελτιώσει εντυπωσιακά τους χρόνους μιας και θα μπορεί να ελέγξει εύκολα και γρήγορα την λειτουργικότητα για κάθε web browser.

Η συνγχώνευση με ένα test management system όπως για παράδειγμα το Squash Test management θα ήταν επίσης μία πολύ αποτελεσματική εξέλιξη του συστήματος. Θα προσφέρει μεγαλύτερη ταχύτητα και ευκολία στην επανεξέταση λαθών και τεστ καθώς επίσης τα δίνει την δυνατότητα στην καταγραφή των αποτελεσμάτων.

ΑΝΑΦΟΡΕΣ

Official Selenium Documentation: http://docs.seleniumhq.org/docs/03_webdriver.jsp

Selenium Project Depot on Google Code:
https://code.google.com/p/selenium/wiki/GettingStarted

Setup guide from Joe Colantonio:
http://www.joecolantonio.com/2012/11/30/selenium-2-0-webdriver-how-to-get-started-with-eclipse-java-ie/

Acceptance testing: http://www.extremeprogramming.org/rules/functionaltests.html

Squash Test Management: http://www.squashtest.org

ΒΙΒΛΙΟΓΡΑΦΙΑ

Cedric Beust, Hani Suleiman - Next Generation Java Testing

Selenium Documentation

Myers, Glenford (2004). *The Art of Software Testing*. Wiley. ISBN 978-0-471-46912-4.

Testing Experience

Black, Rex (August 2009). Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. Hoboken, NJ: Wiley. ISBN 0-470-40415-9.

Cimperman, Rob (2006). UAT Defined: A Guide to Practical User Acceptance Testing. Pearson Education. pp. Chapter 2. ISBN 9780132702621.

Goethem, Brian Hambling, Pauline van (2013). User acceptance testing : a step-by-step guide. BCS Learning & Development Limited. ISBN 9781780171678.

Pusuluri, Nageshwar Rao (2006). Software Testing Concepts And Tools. Dreamtech Press. p. 62. ISBN 9788177227123.