

ΤΕΙ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ Η/Υ ΣΥΣΤΗΜΑΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΗΧΑΝΙΚΗΣ ΛΟΓΙΣΜΙΚΟΥ

Τελικό Παραδοτέο για το
έργο:

*Online Σύστημα κρατήσεων
ενοικιαζόμενων δωματίων*



Hotel Alma Libre

Ομάδα 11

Αικατερίνη Μαρτίνη
Α.Μ. 38199
Τρίτη 10:00 – 12:00

Ερμιόνη Νικολακάκου
Α.Μ. 37333
Δευτέρα 18:00 – 20:00

Τελικό Παραδοτέο

για την εργασία

Online σύστημα κρατήσεων ενοικιαζόμενων δωματίων

Από την Ομάδα 11

1. Αικατερίνη Μαρτίνη
Α.Μ. 38199
Τρίτη 10:00 – 12:00
2. Ερμιόνη Νικολακάκου
Α.Μ. 37333
Δευτέρα 18:00 – 20:00

Hotel Alma Libre

Αναθεωρήσεις

Συγγραφέας	Ημερομηνία	Αιτία αναθεωρήσης	Έκδοση
Αικατερίνη Μαρτίνη	10/1/2013	Αρχικό πρόχειρο έγγραφο	1.0 draft 1
Ερμιόνη Νικολακάκου	21/01/2013	Γενική αναθεώρηση	1.0 Εγκεκριμένο

Πίνακας Περιεχομένων

1.	Εισαγωγή.....	7
1.1.	Σκοπός του παρόντος κειμένου	7
1.2.	Πεδίο δράσης του Έργου	7
1.3.	Περίληπτική παρουσίαση του παρόντος κειμένου	7
1.4	Αναφορές	7
2	Γενική Περιγραφή Συστήματος	8
2.1	Διαγράμματα Περιπτώσεων χρήσης.....	8
2.1.1	BookRoomUseCaseDiagram	8
2.1.2	CheckInUseCaseDiagram	9
2.1.3	CheckOutUseCaseDiagram	9
2.2	Συνοπτική περιγραφή των εμπλεκόμενων ρόλων.....	10
2.3	Συνοπτική περιγραφή των περιπτώσεων χρήσης.....	10
3.	Λειτουργικές απαιτήσεις – Use cases	11
3.1	Αναζήτηση διαθεσιμότητας δωματίων - Υπάλληλος.....	11
3.2	Καταχώρηση νέας κράτησης - Υπάλληλος.....	12
3.3	Check-in δωματίου.....	13
3.4	Check-out δωματίου	14
3.5	Ενημέρωση λίστας εξόδων	15
3.6	Εκτύπωση Απόδειξης	16
3.7	Καταχώρηση/ Ενημέρωση Προσφορών	17
3.8	Εκτύπωση Στατιστικών.....	18
3.9	Αναζήτηση διαθεσιμότητας δωματίων - Πελάτης.....	19
3.10	Κράτηση δωματίου - Πελάτης.....	20
4	Μη λειτουργικές απαιτήσεις	21
4.1	Περιβάλλον λειτουργίας	21
4.2	Περιορισμοί στη σχεδίαση και την υλοποίηση	21
4.3	Προϋποθέσεις - Εξαρτήσεις.....	21
4.4	Απαιτήσεις για τις εξωτερικές διεπαφές	22
4.4.1	Διεπαφή χρήστη	22
4.4.2	Διεπαφές υλικού	24
4.4.3	Διεπαφές λογισμικού.....	24
4.4.4	Διεπαφές επικοινωνιών	24
4.5	Επιδόσεις.....	25
4.6	Φυσική ασφάλεια	25
4.7	Ασφάλεια πληροφορίας	25
4.8	Ποιότητα λογισμικού	25
4.9	Επιχειρησιακοί κανόνες	25
5	Περιγραφή δεδομένων	26
5.1	Σχεσιακό διάγραμμα	26
5.3	Επεξήγηση πινάκων	26
6	Τεκμηρίωση για τον χρήστη	27
7	Διάγραμμα κλάσεων	29
8	Διαγράμματα ακολουθίας	30
9	Διαγράμματα δραστηριοτήτων.....	33
10	Αποτελέσματα static analysis PMD πριν και μετά	34
10.1	PMD πριν.....	34
10.2	PMD μετά	34
11	Μέθοδοι Unit esting.....	37
12	Πηγαίος κώδικας – Screenshots – GUI	41

12.1 GUI CODE.....	41
12.2 Model CODE	59
12.3 Screenshots	77
13 User Acceptance Testing.....	84
Ορισμός Ελέγχου Αποδοχής χρηστών	84
Υπεύθυνοι Ελέγχου Αποδοχής Χρηστών	84
Έλεγχος.....	84

Καταγραφή & Ανάλυση Απαιτήσεων

1. Εισαγωγή

1.1. Σκοπός του παρόντος κειμένου

Αυτό το έγγραφο προδιαγραφών απαιτήσεων λογισμικού περιγράφει τις λειτουργικές και τις μη λειτουργικές απαιτήσεις της έκδοσης 1.0 του Online συστήματος κρατήσεων ενοικιαζόμενων δωματίων για την εταιρεία Alma libre hostel. Το παρόν έγγραφο προορίζεται να χρησιμοποιηθεί από τα μέλη του έργου τα οποία θα υλοποιήσουν και θα επιβεβαιώσουν την ορθή λειτουργία του συστήματος. Αν δεν περιγράφεται ρητώς με άλλο τρόπο, όλες οι απαιτήσεις που αναφέρονται στο παρόν έγγραφο είναι υψηλής προτεραιότητας και απαραίτητες για την πρώτη έκδοση του προγράμματος.

1.2. Πεδίο δράσης του Έργου

Το Online σύστημα κρατήσεων ενοικιαζόμενων δωματίων πρόκειται να υλοποιηθεί ως ένα **νέο λογισμικό** το οποίο αναλόγως με τις ανάγκες του πελάτη πρόκειται να επεκταθεί σταδιακά. Αυτό το λογισμικό θα αναλάβει να αυτοματοποιήσει τη διαδικασία κράτησης δωματίων και τήρησης πελατολογίου για το hostel Alma Libre η οποία μέχρι πρότινος γινόταν χειροκίνητα. Αυτή η κίνηση εναρμονίζεται με τους στρατηγικούς στόχους της επιχείρησης καθώς άμεσο αποτέλεσμα θα είναι η αύξηση της παραγωγικότητας του ανθρωπίνου δυναμικού και η αύξηση της ικανοποίησης των πελατών που θα απολαμβάνουν μια αυτοματοποιημένη διαδικασία κράτησης δωματίου. Επίσης θα ελαχιστοποιηθούν τα λάθη τα οποία κοστίζουν χρόνο και χρήμα και μέσα από το σύστημα θα επιτευχθεί η βέλτιστη ανάλυση των προηγούμενων ετών καθώς η προσομοίωση μελλοντικών καταστάσεων που θα βοηθήσουν στην λήψη της καλύτερης απόφασης για το μέλλον της επιχείρησης. Ανάμεσα στις δυνατότητες του θα συμπεριληφθεί η δυνατότητα υποστήριξης του πελάτη σε μελλοντική επέκτασή του. (πχ. Δημιουργία νέου ξενοδοχείου ή μεγάλη αύξηση ρόλων.).

1.3. Περιληπτική παρουσίαση του παρόντος κειμένου

Το παρόν έγγραφο αποτελεί τη βάση υλοποίησης της εφαρμογής καθώς περιλαμβάνει όλες τις απαιτήσεις και οδηγίες που χρειάζονται προκειμένου να επιτευχθεί η ικανοποίηση των αναγκών του πελάτη. Το κείμενο προορίζεται για ανάγνωση από τον Ιδιοκτήτη της επιχείρησης καθώς και τους υπαλλήλους του γι αυτό το λόγο η δομή του είναι απλή. Όλες οι επιχειρηματικές ανάγκες του πελάτη έχουν μεταφραστεί σε μοντέλα που θα βοηθήσουν τους προγραμματιστές να υλοποιήσουν την προδιαγραφμένη εφαρμογή.

1.4 Αναφορές

1. Καταστατικό εταιρίας
2. Λογιστικό βιβλίο

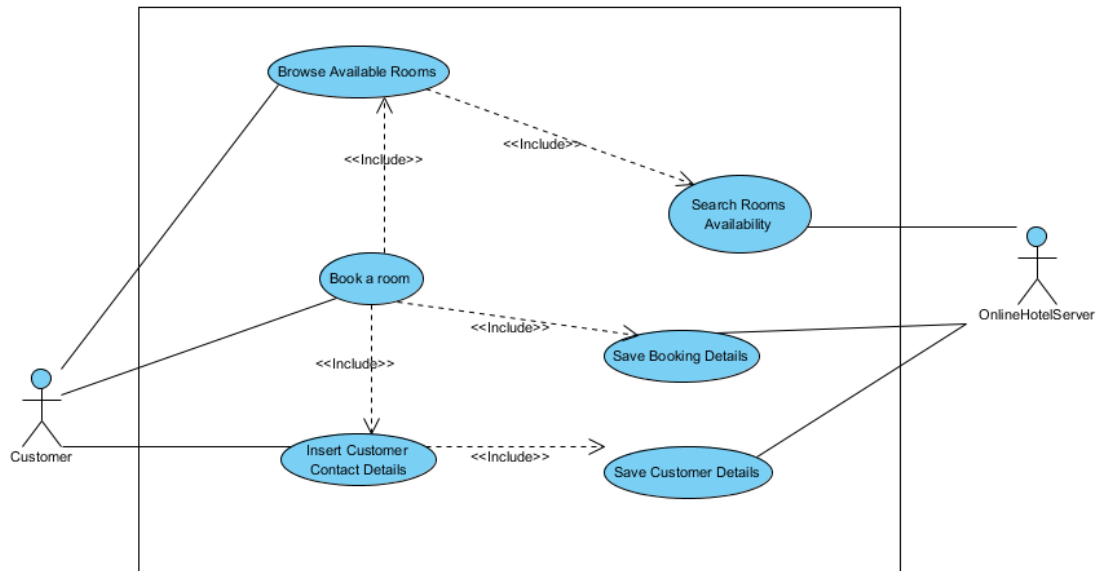
2 Γενική Περιγραφή Συστήματος

Το σύστημα που θα υλοποιήσουμε είναι μια απλή Online εφαρμογή κράτησης δωματίων η οποία θα υλοποιεί τις παρακάτω λειτουργίες:

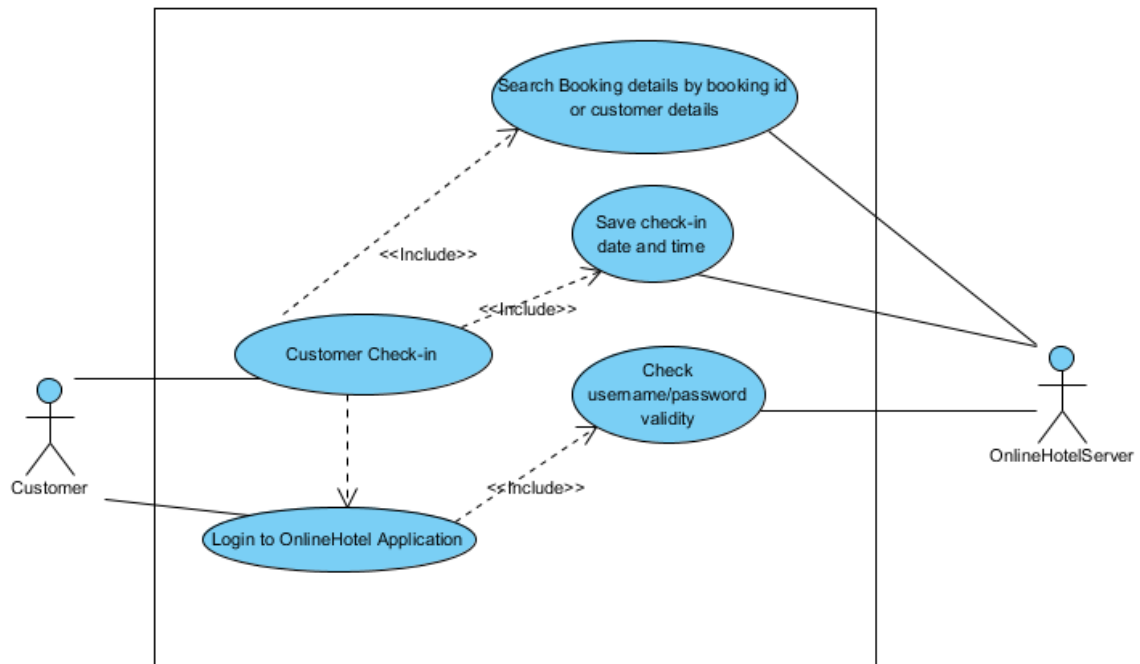
- ☐ Αναζήτηση διαθεσιμότητας δωματίων (πελάτης, υπάλληλος)
- ☐ Κράτηση δωματίου (πελάτης, υπάλληλος)
- ☐ Διαχείριση δωματίων (υπάλληλος)
- ☐ Διαχείριση προσφορών (ιδιοκτήτης)
- ☐ Λίστα εξόδων / Εκτύπωση απόδειξης
- ☐ Στατιστικά (ιδιοκτήτης)

2.1 Διαγράμματα Περιπτώσεων χρήσης

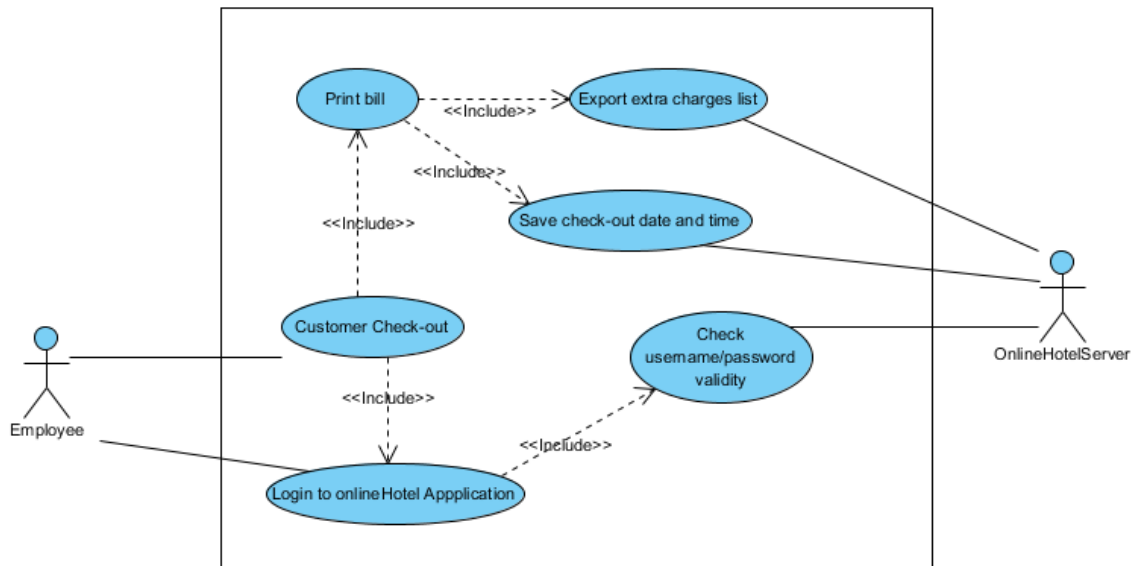
2.1.1 BookRoomUseCaseDiagram



2.1.2 CheckInUseCaseDiagram



2.1.3 CheckOutUseCaseDiagram



2.2 Συνοπτική περιγραφή των εμπλεκόμενων ρόλων

Στο σύστημά μας υπάρχουν τρεις (3) κατηγορίες χρηστών. Ο πελάτης (Customer), ο υπάλληλος (Employee), και ο Admin που στην περίπτωση μας είναι ο Ιδιοκτήτης.

Πελάτης – Customer: Χρήστης περιορισμένης πρόσβασης. Μπορεί να κάνει μόνο αναζήτηση και κράτηση δωματίου.

Υπάλληλος – Employee: Χρήστης μεσαίας πρόσβασης. Μπορεί κάνει αναζήτηση και κράτηση δωματίων καθώς και να αλλάξει παραμέτρους που αφορούν δωμάτια, στοιχεία πελατών και κρατήσεις.

Ιδιοκτήτης – Admin: Υψηλού επιπέδου χρήστης ο οποίος έχει πλήρη έλεγχο της εφαρμογής αλλά και των στοιχείων των χρηστών. Μπορεί να τροποποιήσει όλες τις μεταβλητές της εφαρμογής καθώς και να εξάγει συγκεντρωτικά αποτελέσματα και στατιστικά.

2.3 Συνοπτική περιγραφή των περιπτώσεων χρήσης

<i>Εμπλεκόμενοι Ρόλοι</i>	<i>Περιπτώσεις Χρήσης</i>
Υπάλληλος	UC-1. Αναζήτηση διαθεσιμότητας δωματίων UC-2. Καταχώρηση νέας κράτησης UC-3. Check-in Δωματίου UC-4. Check-out Δωματίου UC-5. Ενημέρωση λίστας εξόδων UC-6. Εκτύπωση Απόδειξης
Ιδιοκτήτης	UC-7. Καταχώρηση/ Ενημέρωση Προσφορών UC-8. Εκτύπωση Στατιστικών
Πελάτης	UC-9. Αναζήτηση διαθεσιμότητας δωματίων UC-10.Κράτηση δωματίου

3. Λειτουργικές απαιτήσεις – Use cases

3.1 Αναζήτηση διαθεσιμότητας δωματίων - Υπάλληλος

Κωδικός Περίπτωσης:	UC-1	
Ονομασία:	Αναζήτηση διαθεσιμότητας δωματίων	
Δημιουργήθηκε από:	Αικατερίνη Μαρτίνη	Τελευταία ενημέρωση από: Αικατερίνη Μαρτίνη
Ημερομηνία Συγγραφής:	15/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 15/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο υπάλληλος χρησιμοποιεί την φόρμα αναζήτησης της εφαρμογής προκειμένου να δει ποια δωμάτια είναι διαθέσιμα.	
Γεγονός Εκκίνησης:	Ο Υπάλληλος αφού κάνει login στο σύστημα, καταχωρεί τα κριτήρια αναζήτησης (διάστημα και αριθμό κρεβατιών) και εκτελεί την αναζήτηση στην βάση του συστήματος	
Προϋποθέσεις:	Έγκυρος κωδικός χρήστη και σωστή συμπλήρωση της φόρμας αναζήτησης.	
Τελική Κατάσταση:	Προβολή των αποτελεσμάτων της αναζήτησης	
Φυσιολογική Ροή:	Login στο σύστημα Επιλογή λειτουργίας αναζήτησης δωματίου Συμπλήρωση της σχετικής φόρμας με τα κριτήρια που απαιτούνται	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	100 χρήσεις ανά ημέρα	
Business Rules:	Ταξινόμηση αποτελεσμάτων αναζήτησης από το φθηνότερο προς το ακριβότερο δωμάτιο	
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος αναζήτησης θα πρέπει να είναι 5 sec	
Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα. Γνώση χρήσης του συστήματος από τους υπαλλήλους	
Σημειώσεις και ζητήματα:		

3.2 Καταχώρηση νέας κράτησης - Υπάλληλος

Κωδικός Περίπτωσης:	UC-2.	
Ονομασία:	Καταχώρηση νέας κράτησης	
Δημιουργήθηκε από:	Αικατερίνη Μαρτίνη	Τελευταία ενημέρωση από: Αικατερίνη Μαρτίνη
Ημερομηνία Συγγραφής:	15/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 15/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο Υπάλληλος αφού κάνει login και ελέγξει την διαθεσιμότητα των δωματίων κάνει κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης.	
Γεγονός Εκκίνησης:	Ο Υπάλληλος αφού κάνει login και ελέγξει την διαθεσιμότητα των δωματίων, επιλέγει να κάνει κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης.	
Προϋποθέσεις:	Υπάρχει διαθεσιμότητα δωματίων	
Τελική Κατάσταση:	Κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης	
Φυσιολογική Ροή:	Login στο σύστημα Επιλογή λειτουργίας αναζήτησης δωματίου Συμπλήρωση της σχετικής φόρμας με τα κριτήρια που απαιτούνται Κράτηση ενός δωματίου	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης 3. Δεν υπάρχει διαθεσιμότητα	
Ενσωματώνει:	UC-1	
Προτεραιότητα:	1	
Συχνότητα χρήσης:	100 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος αναζήτησης θα πρέπει να είναι 10 sec	
Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα. Γνώση χρήσης του συστήματος από τους υπαλλήλους	

3.3 Check-in δωματίου

Κωδικός Περίπτωσης:	UC-3.	
Ονομασία:	Check-in δωματίου	
Δημιουργήθηκε από:	Αικατερίνη Μαρτίνη	Τελευταία ενημέρωση από: Αικατερίνη Μαρτίνη
Ημερομηνία Συγγραφής:	15/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 15/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο Υπάλληλος ελέγχει τα στοιχεία του πελάτη για την ολοκλήρωση του check-in.	
Γεγονός Εκκίνησης:	Ο Υπάλληλος αφού κάνει login επιλέγει να αναζητήσει την κράτηση με βάση τα στοιχεία του πελάτη.	
Προϋποθέσεις:	Να έχει γίνει κράτηση δωματίου για την μέρα που ο πελάτης κάνει check-in	
Τελική Κατάσταση:	Εμφανίζονται στην οθόνη τα στοιχεία του πελάτη και ο αριθμός του δωματίου της κράτησης	
Φυσιολογική Ροή:	Login στο σύστημα Επιλογή λειτουργίας αναζήτησης πελάτη Συμπλήρωση της σχετικής φόρμας με τα στοιχεία του πελάτη Ολοκλήρωση του check-in για τον πελάτη	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	10 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την ολοκλήρωση του check-in δε θα πρέπει να ξεπερνά τα 5 min	
Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα.	

	Γνώση χρήσης του συστήματος από τους υπαλλήλους
Σημειώσεις και ζητήματα:	

3.4 Check-out δωματίου

Κωδικός Περίπτωσης:	UC-4	
Ονομασία:	Check-out δωματίου	
Δημιουργήθηκε από:	Αικατερίνη Μαρτίνη	Τελευταία ενημέρωση από: Αικατερίνη Μαρτίνη
Ημερομηνία Συγγραφής:	15/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 15/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο Υπάλληλος κάνει αναζήτηση της κράτησης με βάση το όνομα του πελάτη ή τον αριθμό του δωματίου του	
Γεγονός Εκκίνησης:	Ο Υπάλληλος αφού κάνει login επιλέγει να αναζητήσει την κράτηση με βάση το όνομα του πελάτη ή τον αριθμό του δωματίου του.	
Προϋποθέσεις:	Να έχει προηγηθεί το check-in του πελάτη	
Τελική Κατάσταση:	Εκτυπώνεται η απόδειξη για τον πελάτη	
Φυσιολογική Ροή:	Login στο σύστημα Επιλογή λειτουργίας αναζήτησης δωματίου Αίτημα έκδοσης απόδειξης Ολοκλήρωση του check-out για τον πελάτη	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	10 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την ολοκλήρωση του check-out δε θα πρέπει να ξεπερνά τα 5 min	
Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα.	

	Γνώση χρήσης του συστήματος από τους υπαλλήλους
Σημειώσεις και ζητήματα:	

3.5 Ενημέρωση λίστας εξόδων

Κωδικός Περίπτωσης:	UC-5.	
Ονομασία:	Ενημέρωση λίστας εξόδων	
Δημιουργήθηκε από:	Αικατερίνη Μαρτίνη	Τελευταία ενημέρωση από: Αικατερίνη Μαρτίνη
Ημερομηνία Συγγραφής:	15/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 15/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο Υπάλληλος κάνει αναζήτηση του δωματίου και καταχωρεί έξοδα που αφορούν τον αντίστοιχο πελάτη	
Γεγονός Εκκίνησης:	Ο Υπάλληλος αφού κάνει login κάνει αναζήτηση του δωματίου και καταχωρεί τυχόν χρεώσεις που αφορούν το δωμάτιο	
Προϋποθέσεις:	Να έχει γίνει το check-in για το συγκεκριμένο δωμάτιο	
Τελική Κατάσταση:	Δημιουργούνται οι απαραίτητες εγγραφές εξόδων στη βάση	
Φυσιολογική Ροή:	1. Login στο σύστημα 2. Επιλογή λειτουργίας αναζήτησης δωματίου με βάση τον αριθμό δωματίου 3. Καταχώρηση των εγγραφών που αντιστοιχούν στα έξοδα	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	10 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την ολοκλήρωση του check-in δε θα πρέπει να ξεπερνά τα 5 min	
Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα.	

	Γνώση χρήσης του συστήματος από τους υπαλλήλους
Σημειώσεις και ζητήματα:	

3.6 Εκτύπωση Απόδειξης

Κωδικός Περίπτωσης:	UC-6	
Ονομασία:	Εκτύπωση Απόδειξης	
Δημιουργήθηκε από:	Ερμιόνη Νικολακάκου	Τελευταία ενημέρωση από: Ερμιόνη Νικολακάκου
Ημερομηνία Συγγραφής:	19/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 19/1/2013
Εμπλεκόμενοι Ρόλοι:	Υπάλληλος	
Περιγραφή:	Ο Υπάλληλος εισάγει τον αριθμό δωματίου και τυπώνει την απόδειξη	
Γεγονός Εκκίνησης:	Ο Υπάλληλος επιλέγει την λειτουργία εκτύπωσης αποδείξεων	
Προϋποθέσεις:	Να έχει γίνει το check-in για το συγκεκριμένο δωμάτιο	
Τελική Κατάσταση:	Δημιουργείται το pdf της απόδειξης που περιλαμβάνει την χρέωση του δωματίου και την λίστα των επιπλέον εξόδων	
Φυσιολογική Ροή:	1. Login στο σύστημα 2. Επιλογή λειτουργίας αναζήτησης δωματίου με βάση τον αριθμό δωματίου 3. Ολοκλήρωση διαδικασίας check-out πελάτη 4. Επιλογή παραγωγής του pdf της απόδειξης	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Λανθασμένα στοιχεία για το login 2. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:	UC-4	
Προτεραιότητα:	1	
Συχνότητα χρήσης:	10 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την παραγωγή του pdf της απόδειξης δε θα πρέπει να είναι μεγαλύτερος από 0.5 min	

Υποθέσεις:	Όλοι οι υπάλληλοι έχουν username και password που τους επιτρέπει να κάνουν login στο σύστημα. Γνώση χρήσης του συστήματος από τους υπαλλήλους
Σημειώσεις και ζητήματα:	

3.7 Καταχώρηση/ Ενημέρωση Προσφορών

Κωδικός Περίπτωσης:	UC-7	
Ονομασία:	Καταχώρηση/ Ενημέρωση Προσφορών	
Δημιουργήθηκε από:	Ερμιόνη Νικολακάκου	Τελευταία ενημέρωση από: Ερμιόνη Νικολακάκου
Ημερομηνία Συγγραφής:	19/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 19/1/2013
Εμπλεκόμενοι Ρόλοι:	Ιδιοκτήτης	
Περιγραφή:	Ο ιδιοκτήτης καταχωρεί τις προσφορές με την μορφή ποσοστού έκπτωσης επί της τιμής του δωματίου	
Γεγονός Εκκίνησης:	Ο ιδιοκτήτης αφού κάνει login ανοίγει την φόρμα καταχώρησης προσφορών	
Προϋποθέσεις:	Ο ιδιοκτήτης έχει ενεργό κωδικό και όνομα χρήστη	
Τελική Κατάσταση:	Η νέα τιμή της έκπτωσης καταχωρείται στη βάση.	
Φυσιολογική Ροή:	<ol style="list-style-type: none"> 1. Ο ιδιοκτήτης κάνει login 2. Ανοίγει την φόρμα καταχώρησης/ενημέρωσης προσφορών 3. Αποθηκεύει την τιμή της έκπτωσης στην βάση 	
Εναλλακτική Ροή:		
Εξαιρέσεις:	<ol style="list-style-type: none"> 1. Δεν είναι έγκυρο το όνομα χρήστη ή/και ο κωδικός εισόδου στην εφαρμογή 2. Δεν είναι έγκυρη η τιμή έκπτωσης που έχει καταχωρηθεί 	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	2 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την αποθήκευση/ενημέρωση της βάσης	

	με την έκπτωση δε θα πρέπει να είναι μεγαλύτερος από 5 sec
Υποθέσεις:	Ο ιδιοκτήτης έχει username και password που του επιτρέπει να κάνει login στο σύστημα με αυξημένα δικαιώματα.
Σημειώσεις και ζητήματα:	

3.8 Εκτύπωση Στατιστικών

Κωδικός Περίπτωσης:	UC-8	
Ονομασία:	Εκτύπωση Στατιστικών	
Δημιουργήθηκε από:	Ερμιόνη Νικολακάκου	Τελευταία ενημέρωση από: Ερμιόνη Νικολακάκου
Ημερομηνία Συγγραφής:	19/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 19/1/2013
Εμπλεκόμενοι Ρόλοι:	Ο Ιδιοκτήτης	
Περιγραφή:	Ο ιδιοκτήτης επιλέγει την προβολή στατιστικών στοιχείων που αφορούν στο πλήθος των κρατήσεων σε συγκεκριμένες χρονικές περιόδους.	
Γεγονός Εκκίνησης:	Ο Ιδιοκτήτης αφού κάνει login ανοίγει την φόρμα εκτύπωσης στατιστικών.	
Προϋποθέσεις:	Ο ιδιοκτήτης έχει ενεργό κωδικό και όνομα χρήστη	
Τελική Κατάσταση:	Εκτυπώνονται στην οθόνη ο αριθμός των κρατήσεων που έγιναν στην υπό εξέταση χρονική περίοδο καθώς και η λίστα με τα αναλυτικά στοιχεία των κρατήσεων	
Φυσιολογική Ροή:	<ol style="list-style-type: none"> 1. Ο ιδιοκτήτης κάνει login 2. Ανοίγει την φόρμα εκτύπωσης στατιστικών κράτησης 3. Εισάγει την έναρξη και λήξη της υπο εξέταση χρονικής περιόδου 4. Εκτυπώνονται στην οθόνη τα στατιστικά για τις κρατήσεις που έγιναν την χρονική περίοδο που είχε καταχωρηθεί στο βήμα 3 	
Εναλλακτική Ροή:		
Εξαιρέσεις:	<ol style="list-style-type: none"> 1. Δεν είναι έγκυρο το όνομα χρήστη ή/και ο κωδικός εισόδου στην εφαρμογή 2. Δεν είναι έγκυρο το διάστημα που έχει καταχωρηθεί 	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	2 χρήσεις ανά ημέρα	
Business Rules:		

Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος για την εκτύπωση των στατιστικών δε θα πρέπει να είναι μεγαλύτερος από 0.5 min
Υποθέσεις:	Ο ιδιοκτήτης έχει username και password που του επιτρέπει να κάνει login στο σύστημα με αυξημένα δικαιώματα.
Σημειώσεις και ζητήματα:	

3.9 Αναζήτηση διαθεσιμότητας δωματίων - Πελάτης

Κωδικός Περίπτωσης:	UC-9	
Ονομασία:	Αναζήτηση διαθεσιμότητας δωματίων	
Δημιουργήθηκε από:	Ερμιόνη Νικολακάκου	Τελευταία ενημέρωση από: Ερμιόνη Νικολακάκου
Ημερομηνία Συγγραφής:	19/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 19/1/2013
Εμπλεκόμενοι Ρόλοι:	Πελάτης	
Περιγραφή:	Ο πελάτης χρησιμοποιεί την φόρμα αναζήτησης της εφαρμογής προκειμένου να δει ποια δωμάτια είναι διαθέσιμα.	
Γεγονός Εκκίνησης:	Ο πελάτης καταχωρεί τα κριτήρια αναζήτησης (διάστημα και αριθμό κρεβατιών) και εκτελεί την αναζήτηση στην βάση του συστήματος	
Προϋποθέσεις:	Σωστή συμπλήρωση της φόρμας αναζήτησης.	
Τελική Κατάσταση:	Προβολή των αποτελεσμάτων της αναζήτησης	
Φυσιολογική Ροή:	Επιλογή λειτουργίας αναζήτησης δωματίου Συμπλήρωση της σχετικής φόρμας με τα κριτήρια που απαιτούνται	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. εσφαλμένα κριτήρια αναζήτησης	
Ενσωματώνει:		
Προτεραιότητα:	1	
Συχνότητα χρήσης:	100 χρήσεις ανά ημέρα	
Business Rules:	Ταξινόμηση αποτελεσμάτων αναζήτησης από το φθηνότερο προς το ακριβότερο δωμάτιο	
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος αναζήτησης θα πρέπει να είναι 5 sec	
Υποθέσεις:		
Σημειώσεις και ζητήματα:		

3.10 Κράτηση δωματίου - Πελάτης

Κωδικός Περίπτωσης:	UC-10	
Ονομασία:	Κράτηση δωματίου	
Δημιουργήθηκε από:	Ερμιόνη Νικολακάκου	Τελευταία ενημέρωση από: Ερμιόνη Νικολακάκου
Ημερομηνία Συγγραφής:	19/1/2013	Ημερομηνία τελευταίας ενημέρωσης: 19/1/2013
Εμπλεκόμενοι Ρόλοι:	Πελάτης	
Περιγραφή:	Ο Πελάτης αφού ελέγξει την διαθεσιμότητα των δωματίων κάνει κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης.	
Γεγονός Εκκίνησης:	Ο Πελάτης αφού ελέγξει την διαθεσιμότητα των δωματίων, επιλέγει να κάνει κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης.	
Προϋποθέσεις:	Υπάρχει διαθεσιμότητα δωματίων	
Τελική Κατάσταση:	Κράτηση ενός δωματίου που ικανοποιεί τα κριτήρια αναζήτησης	
Φυσιολογική Ροή:	Επιλογή λειτουργίας αναζήτησης δωματίου Συμπλήρωση της σχετικής φόρμας με τα κριτήρια που απαιτούνται Κράτηση ενός δωματίου	
Εναλλακτική Ροή:		
Εξαιρέσεις:	1. Εσφαλμένα κριτήρια αναζήτησης 3. Δεν υπάρχει διαθεσιμότητα	
Ενσωματώνει:	UC-1	
Προτεραιότητα:	1	
Συχνότητα χρήσης:	100 χρήσεις ανά ημέρα	
Business Rules:		
Ειδικές απαιτήσεις:	Ο μέγιστος χρόνος αναζήτησης θα πρέπει να είναι 10 sec	
Υποθέσεις:		
Σημειώσεις και ζητήματα:		

4 Μη λειτουργικές απαιτήσεις

4.1 Περιβάλλον λειτουργίας

ΠΛ-1: Ο client του λογισμικού θα πρέπει να μπορεί να εκτελεστεί σε υπολογιστές με λειτουργικό Windows XP και άνω

4.2 Περιορισμοί στη σχεδίαση και την υλοποίηση

ΠΣ-1: Είναι απαραίτητο να ληφθεί υπόψη ότι ο πελάτης δε διατίθεται να προβεί σε αγορά νέου hardware γι αυτό θα γίνει χρήση της υπάρχουσας υποδομής. Η υπάρχουσα υποδομή αποτελείται από έναν server **Pentium 2.8 GHz, 8GB RAM** και λειτουργικό σύστημα **Linux**. Το λογισμικό ΔΕΝ θα συνυπάρξει με άλλες εφαρμογές πέραν της **MYSQL 5.5**

Έπειτα από εκπαίδευση ο πελάτης θα είναι υπεύθυνος για τη λήψη αντιγράφων ασφαλείας και διατήρησης της εφαρμογής σε λειτουργική κατάσταση και εγκατάσταση νέων εκδόσεων.

ΠΣ-2: Ο πελάτης δεν επιθυμεί επίσης να αγοράσει νέους υπολογιστές. Η εφαρμογή θα πρέπει να είναι ικανή να λειτουργήσει στους υπάρχοντες υπολογιστές οι οποίοι έχουν τα παρακάτω χαρακτηριστικά: **Windows XP, Pentium 2GHz, 4 GB RAM**

ΠΣ-3: Η γλώσσα προγραμματισμού που θα πρέπει να χρησιμοποιήσουμε είναι **Java**

ΠΣ-4: Πρέπει οπωσδήποτε να χρησιμοποιήσουμε μοντέλο σχεσιακής **βάσης δεδομένων**.

ΠΣ-5: Θα πρέπει να ληφθεί υπόψη ότι οι χρήστες (πελάτες) έχουν πρόσβαση στην εφαρμογή μέσω **ADSL** γραμμής τουλάχιστον **2MBPS**.

ΠΣ-6: Θα πρέπει να ληφθεί υπόψη ότι ο ιδιοκτήτης και οι υπάλληλοι έχουν πρόσβαση στην εφαρμογή από δίκτυο **LAN 1GBps**.

4.3 Προϋποθέσεις - Εξαρτήσεις

ΕΞ-1: Η λειτουργία της εφαρμογής εξαρτάται από το κατά πόσο οι χρήστες είναι επιμελείς ως προς την εισαγωγή των στοιχείων στη βάση δεδομένων

ΕΞ-2: Η λειτουργία της εφαρμογής εξαρτάται από το κατά πόσο ενημερώνεται και εναρμονίζεται με νέους νόμους. Είναι υποχρέωση της εταιρείας να επιβεβαιώνει ότι όντως η εφαρμογή ανταποκρίνεται στις νομικές απαιτήσεις.

ΕΞ-3: Το λογισμικό θα πρέπει να είναι σε λειτουργία **24x7x365**

ΕΞ-4: Το λογισμικό θα πρέπει να μπορεί να εξυπηρετεί κάθε συναλλαγή σε χρόνο **<10 sec**

ΕΞ-5: Το λογισμικό στο σύνολό του θα πρέπει να μπορεί να εξυπηρετήσει **100 ασύγχρονους χρήστες** τουλάχιστον.

4.4 Απαιτήσεις για τις εξωτερικές διεπαφές

4.4.1 Διεπαφή χρήστη

Το σύστημα θα πρέπει να υλοποιεί δύο περιβάλλοντα διεπαφών με το χρήστη, ένα σε μορφή **web** και ένα σε **μορφή καρτελών**.

Το Web Interface θα περιλαμβάνει όλες τις λειτουργίες που μπορεί να πραγματοποιήσει ο πελάτης. Το συγκεκριμένο web interface θα πρέπει να ενσωματωθεί σε μια web σελίδα η οποία θα διαφημίζει την επιχείρηση του πελάτη πχ. www.alma-libre.gr. Μεταξύ των πληροφοριών της επιχείρησης θα υπάρχει ένα κουμπί που θα γράφει «**Κάντε Κράτηση**» το οποίο όταν θα επιλέγεται θα οδηγεί τον χρήστη σε μια φόρμα δεδομένων όπου θα πρέπει να πληκτρολογήσει τις πληροφορίες για την κράτησή του (checkin-date, checkout-date, κρεβάτια και τα προσωπικά του στοιχεία).

Το interface που θα εμφανίζεται σε μορφή **καρτελών** θα υλοποιείται μόνο για την εφαρμογή που θα εκτελείται τοπικά στον υπολογιστή του υπαλλήλου ή του ιδιοκτήτη. Γι' αυτές τις δύο κατηγορίες χρηστών δεν υπάρχει web interface.

ΔΧ-1: Η φόρμα στην οποία θα ανακατευθύνεται ο πελάτης πατώντας το κουμπί «Κάντε Κράτηση» θα αποτελείται από πεδία τα οποία θα πρέπει να συμπληρώσει ή drop down menus τα οποία θα πρέπει να επιλέξει. Αφού συμπληρώσει όλα τα πεδία θα υπάρχει ένα κουμπί «**Book**» το οποίο θα καταχωρεί την κράτηση.

Τα drop down menus θα είναι τα:

- Check-in date (Month, Day)
- Check-Out date (Month,Day)
- Num. of beds (2,3,4)
- Select a room (rooms available).

Τα πεδία που θα πρέπει να συμπληρώσει είναι:

- Firstname
- Lastname
- Phone
- Cell Phone
- ID card
- e-mail
- Address.

ΔΧ-2: Η τοπική εφαρμογή αρχικά θα εμφανίζει μια φόρμα που προτρέπει το χρήστη να επιλέξει Ρόλο και να εισάγει Username και password για την μετάβαση στο κυρίως λογισμικό.

ΔX-3: Αν έχει επιλεγθεί ο χρήστης Employee θα εμφανιστεί μια φόρμα η οποία θα έχει τις εξής καρτέλες:

1. **Search/Book Room**
2. **Check-in/Check-out**
3. **Charges**

ΔX-4: Η καρτέλα **Search/Book Room** θα έχει τα ίδια ακριβώς πεδία που έχει και η φόρμα που συμπληρώνει ο χρήστης που μπαίνει από το web interface. Αυτή τη λειτουργία τη θέλουμε έτσι ώστε να μπορούν να γίνουν τηλεφωνικές κρατήσεις με τον παραδοσιακό τρόπο. Η καρτέλα Search/Book Room θα αποτελείται από πεδία τα οποία θα πρέπει να συμπληρώσει ή drop down menus τα οποία θα πρέπει να επιλέξει.

Τα drop down menus θα είναι τα:

- Check-in date (Month, Day)
- Check-Out date (Month,Day)
- Num. of beds (2,3,4)
- Select a room (rooms available).

Τα πεδία που θα πρέπει να συμπληρώσει είναι:

- Firstname
- Lastname
- Phone
- Cell Phone
- ID card
- e-mail
- Address.

ΔX-5: Η καρτέλα **Check-in/Check-out** θα επιβεβαιώνει ουσιαστικά το Checkin και το Checkout.

Ο Employee θα πρέπει να εισάγει το **booking id** και να πατήσει το κουμπί **Check-in** για να γίνει το Check-in.

Ο Employee θα πρέπει να εισάγει το **room num** και να πατήσει το κουμπί **Check-out** για να γίνει το Check-out.

ΔX-6: Η καρτέλα **Charges** υλοποιεί τις χρεώσεις.

Έχει 3 πεδία που θα πρέπει να συμπληρωθούν:

Room num
Description
Price

Πατώντας το κουμπί Add Charge γίνεται η χρέωση

ΔX-7: Αν έχει επιλεγθεί ο χρήστης Admin θα εμφανιστεί μια φόρμα η οποία θα έχει τις εξής καρτέλες:

1. **Offers Management**
2. **Statistics**

ΔX-8: Η καρτέλα **Offers Management** θα έχει ένα dropdown menu με τα δωμάτια (room) που διαχειρίζεται η εφαρμογή και ένα πεδίο Discount (%) στο οποίο θα καταχωρείται το ποσοστό έκπτωσης. Η καταχώρηση γίνεται πατώντας το κουμπί Save Discount.

ΔX-9: Η καρτέλα **Statistics** θα έχει ένα κουμπί **Export Statistics** το οποίο όταν πατιέται θα εξαγει σε μορφή xlsx προκαθορισμένα στατιστικά στοιχεία.

- ΔΧ-10: Το σύστημα θα επικοινωνεί με τον χρήστη με εμφάνιση μηνυμάτων όταν έχουμε:
1. Λανθασμένη εισαγωγή στοιχείων (UserName και/ή PassWord) κατά την είσοδο στην εφαρμογή
 2. Λανθασμένη εισαγωγή στοιχείων Υπαλλήλου (όπως Α.Μ. διπλότυπο)
 3. Μη επιτρεπτή χρήση λειτουργιών
 4. Μη επιτρεπτή πρόσβαση
- ΔΧ-11: Το σύστημα θα παρέχει την δυνατότητα χρήσης πληκτρολογίου και ποντικιού καθ' όλη την χρήση του προγράμματος.
- ΔΧ-12: Το πρόγραμμα θα περιλαμβάνει μενού βοήθειας με ανακατεύθυνση σε σελίδα html που θα παρέχεται πλήρης βοήθεια σχετικά με την εφαρμογή

4.4.2 Διεπαφές υλικού

ΔΥ-1: Το σύστημα θα πρέπει να υλοποιεί διεπαφή με τον εγκατεστημένο εκτυπωτή του υπολογιστή προκειμένου να είναι εφικτή η εκτύπωση στατιστικών και αποδείξεων.

4.4.3 Διεπαφές λογισμικού

ΔΛ-1: Το σύστημα θα πρέπει να υλοποιεί διεπαφή με το σύστημα πληρωμών **EasyBank** προκειμένου οι πελάτες να μπορούν να κάνουν online πληρωμές με ταχύτητα και ασφάλεια

4.4.4 Διεπαφές επικοινωνιών

ΔΕ-1: Θα πρέπει να ληφθεί υπόψη ότι οι χρήστες (πελάτες) έχουν πρόσβαση στην εφαρμογή μέσω **ADSL** γραμμής τουλάχιστον **2MBPS**.

ΔΕ-2: Θα πρέπει να ληφθεί υπόψη ότι ο ιδιοκτήτης και οι υπάλληλοι έχουν πρόσβαση στην εφαρμογή από δίκτυο **LAN 1GBps**.

ΔΕ-3: Θα πρέπει να αγοραστεί ένα domain www.alma-libre.gr το οποίο θα χρησιμοποιηθεί για την επικοινωνία της εφαρμογής με τους πελάτες.

ΔΕ-4: Βάση της ΔΕ-3 η εφαρμογή θα πρέπει να υλοποιεί interfaces που μπορούν να επικοινωνήσουν με πρωτόκολλο **http** ή **https**.

4.5 Επιδόσεις

E-1: Το λογισμικό θα πρέπει να μπορεί να εξυπηρετεί κάθε συναλλαγή σε χρόνο **<10 sec**

E-2: Το λογισμικό στο σύνολό του θα πρέπει να μπορεί να εξυπηρετήσει **100 ασύγχρονους χρήστες** τουλάχιστον.

4.6 Φυσική ασφάλεια

ΦΑ-1: Το σύστημα θα πρέπει να είναι συμμορφωμένο με την αντίστοιχη παράγραφο που αναφέρεται στην φυσική ασφάλεια του **Data Protection Act document.**, το οποίο θα μας παράσχει η εταιρεία .

ΦΑ-2: Το σύστημα θα πρέπει να είναι σε θέση να αποθηκευθεί σε οπτικό δίσκο δεδομένων ή αντίστοιχο μέσο προκειμένου να αποτραπεί πιθανότητα απώλειας δεδομένων. Είναι ευθύνη του πελάτη να εκτελεί τις αντίστοιχες διαδικασίες.

ΦΑ-3: Το σύστημα θα εγκατασταθεί σε ασφαλές χώρο στο **γραφείο του ιδιοκτήτη**.

4.7 Ασφάλεια πληροφορίας

ΑΠ-1: Όλες οι δικτυακές συνεδρίες (sessions) θα πρέπει να είναι κρυπτογραφημένες με **SSL** προκειμένου να διασφαλιστεί το απόρρητο των δεδομένων.

ΑΠ-2: Όλοι οι χρήστες θα πρέπει να ταυτοποιούνται μέσω username και password για όλες τις διαδικασίες.

ΑΠ-3: Ο Διαχειριστής του συστήματος (Administrator) θα έχει δικό του μοναδικό αναγνωριστικό και θα είναι σε θέση μόνο αυτός να προσθέσει νέους χρήστες και να αλλάξει κωδικούς. Επίσης ο Administrator θα μπορεί να κάνει μόνο συγκεντρωτικές εκτυπώσεις αναφορών και καταστάσεων μέσω του προγράμματος.

ΑΠ-4: Οι πελάτες δε θα μπορούν να βλέπουν κρατήσεις άλλων πελατών.

4.8 Ποιότητα λογισμικού

ΠΛ-1: Σημαντικό χαρακτηριστικό του συστήματος είναι η προσαρμοστικότητα. Μια εφαρμογή online κρατήσεων απαιτείται να προσαρμόζεται σε νέες απαιτήσεις..

ΠΛ-2: Είναι απαραίτητο να υλοποιηθεί ικανός αριθμός unit testing που θα επιβεβαιώνει την ποιότητα του κώδικα.

ΠΛ-3: Θα πρέπει να πραγματοποιηθεί stress test του συστήματος με 100 χρήστες.

4.9 Επιχειρησιακοί κανόνες

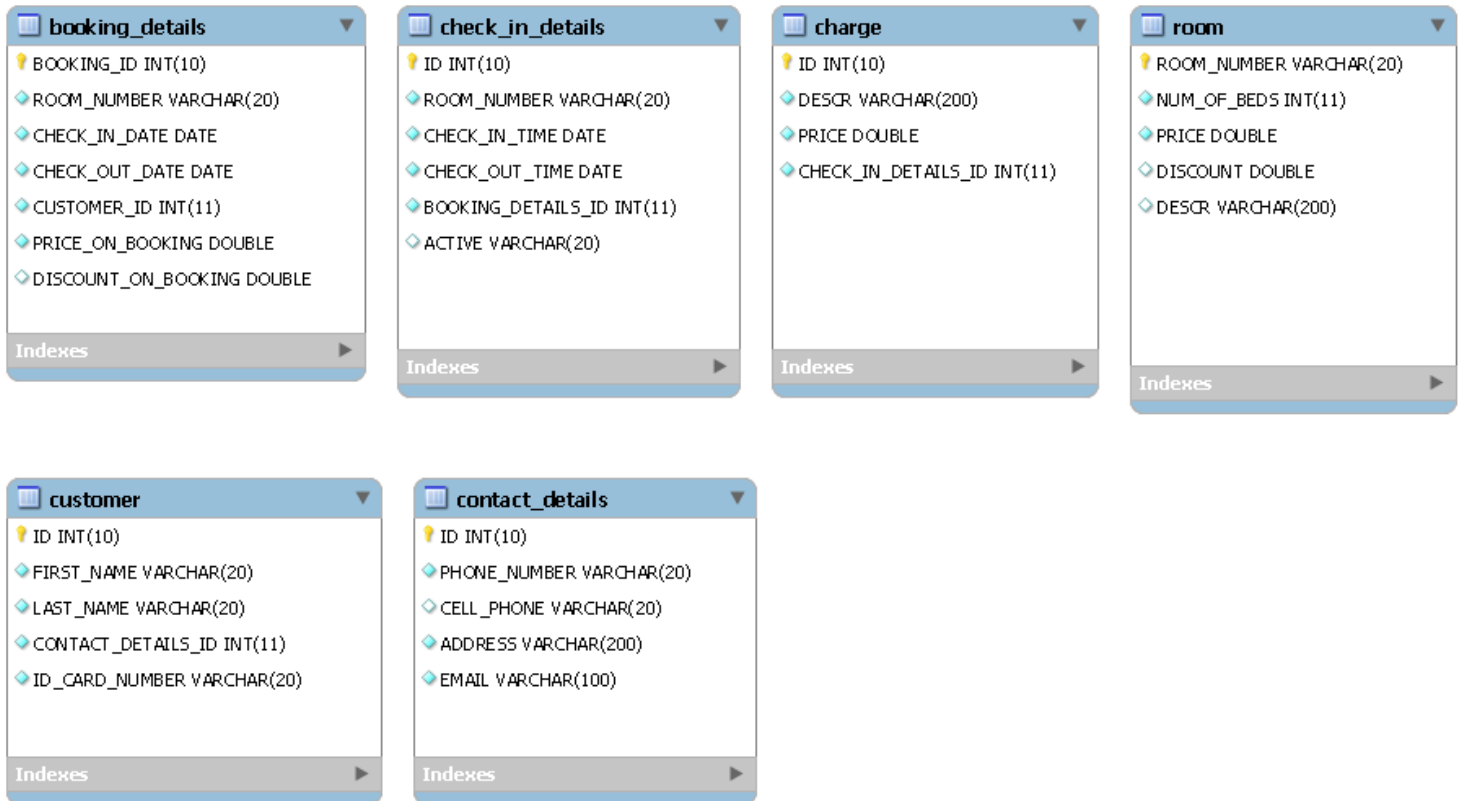
ΕΚ-1: Το σύστημα θα πρέπει να είναι συμμορφωμένο με το **Data Protection Act 1992** το οποίο θα παράσχει ο πελάτης

ΕΚ-2: Το σύστημα θα πρέπει να διαχωρίζει τους χρήστες και να δίνει στον καθένα μόνο τις δυνατότητες που του αναλογούν.

ΕΚ-3: Ο Administrator μόνο μπορεί να εξάγει συγκεντρωτικές καταστάσεις και στατιστικά

5 Περιγραφή δεδομένων

5.1 Σχεσιακό διάγραμμα



5.3 Επεξήγηση πινάκων

Booking_details: περιλαμβάνει τα στοιχεία για κράτηση δωματίου (αριθμός δωματίου, ημερομηνίες, id πελάτη, τιμή, έκπτωση)

Check_in_details: περιλαμβάνει τα στοιχεία για τα checkin (δωμάτιο, ημέρες, λοιπές πληροφορίες)

Charge: Περιλαμβάνει τα στοιχεία για την χρέωση των δωματίων

Room: περιλαμβάνει χαρακτηριστικά των δωματίων (τιμή, αριθμό κρεβατιών, έκπτωση, περιγραφή)

Customer: περιλαμβάνει τα βασικά στοιχεία των πελατών

Contact_details: περιέχει τα στοιχεία επικοινωνίας των πελατών

6 Τεκμηρίωση για τον χρήστη

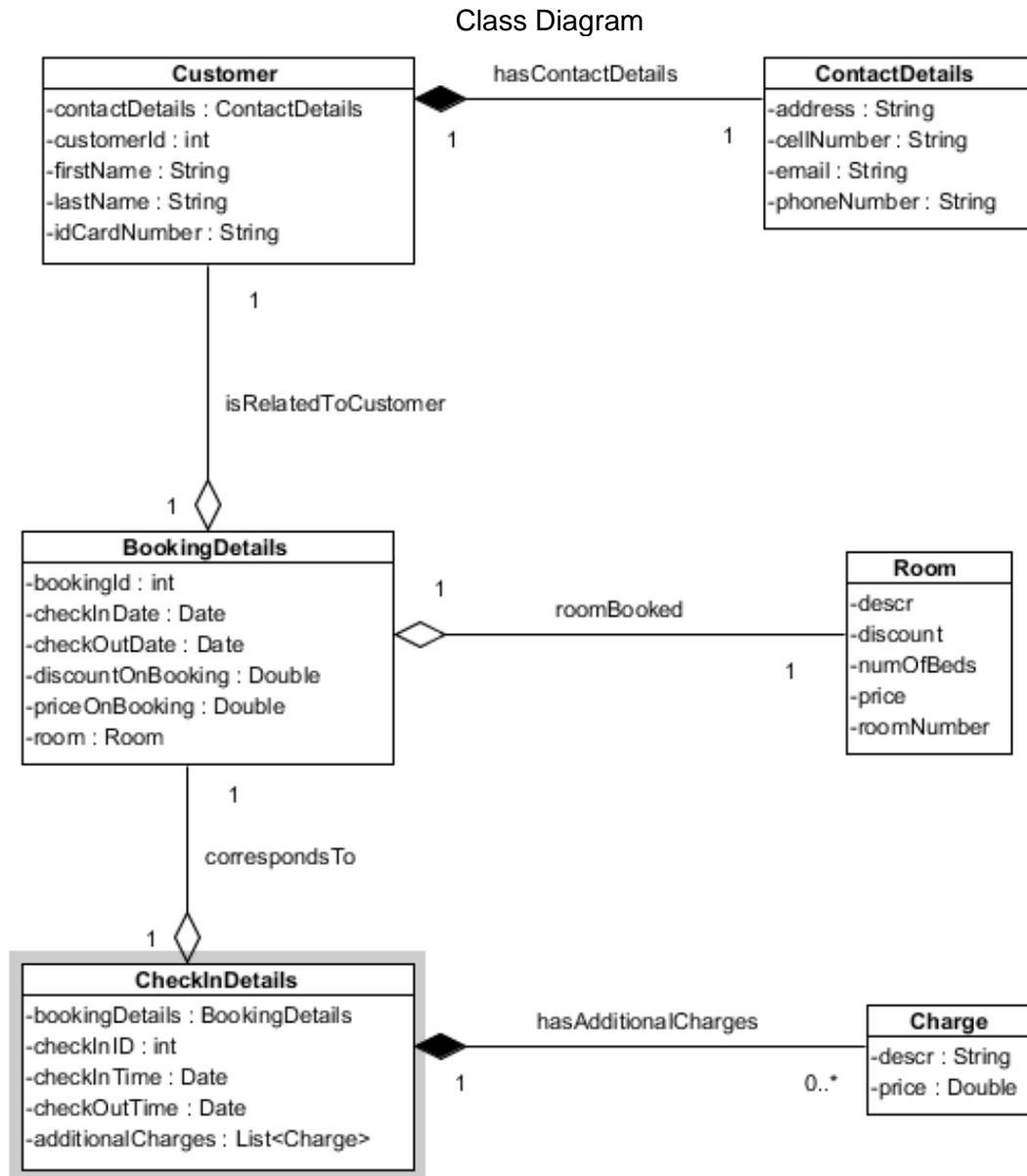
TX-1: Θα παραδοθεί εγχειρίδιο χρήσης του προγράμματος σε έντυπη μορφή το οποίο θα διαφοροποιείται για κάθε τύπο χρήστη και θα είναι σε μεγάλο βαθμό εικονογραφημένο

TX-2: Το σύστημα θα παρέχει τη δυνατότητα βοήθειας στο χρήστη με on-line βοήθεια από το αντίστοιχο μενού.

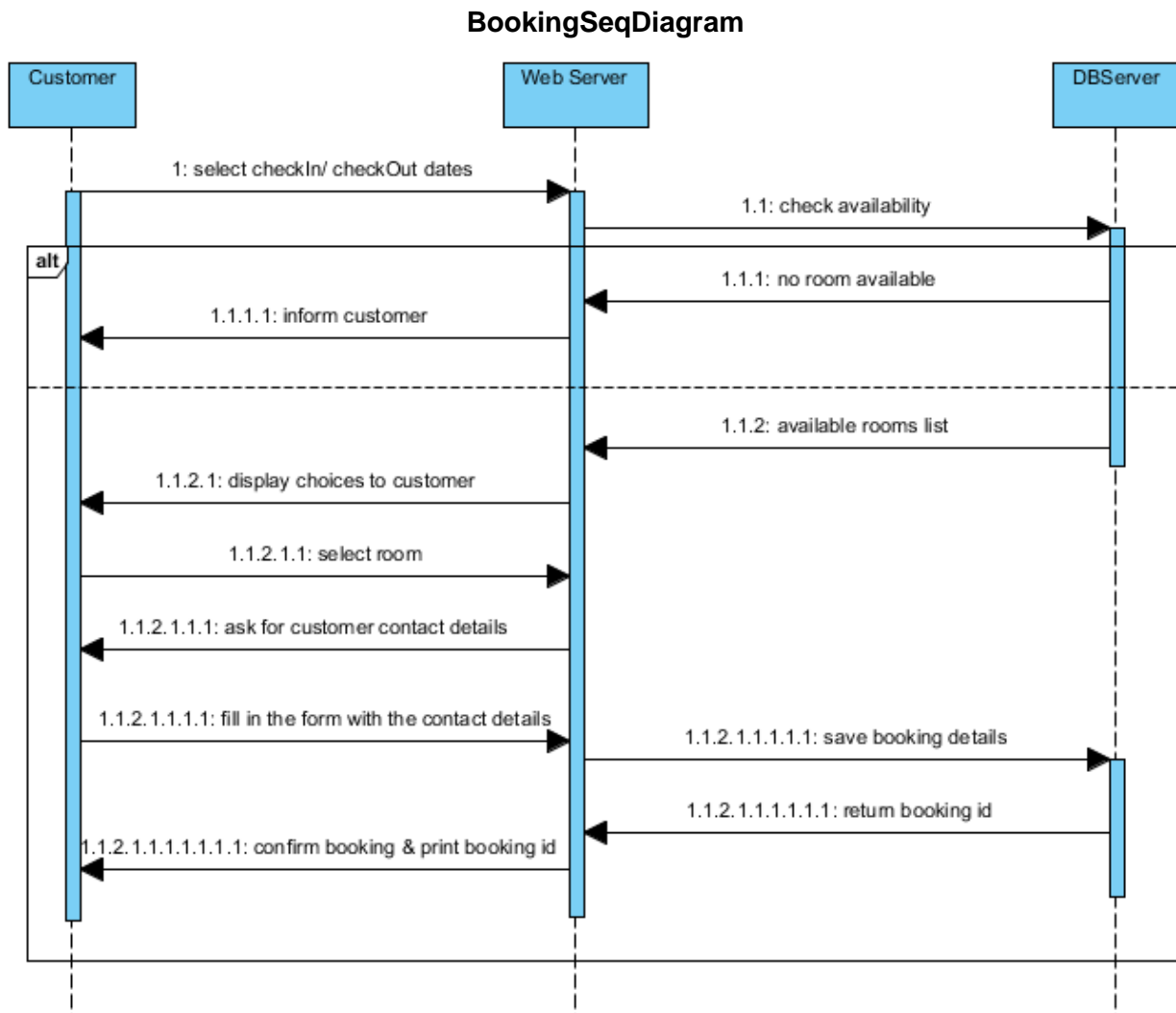
TX-3: Το σύστημα θα περιλαμβάνει και μια εικονική βάση παρόμοια με τη βάση παραγωγής προκειμένου να εξοικειωθούν ο χρήστης και να αποφευχθούν πιθανά προβλήματα στο live σύστημα.

Διαγράμματα Κλάσεων – Δραστηριοτήτων - Ακολουθίας

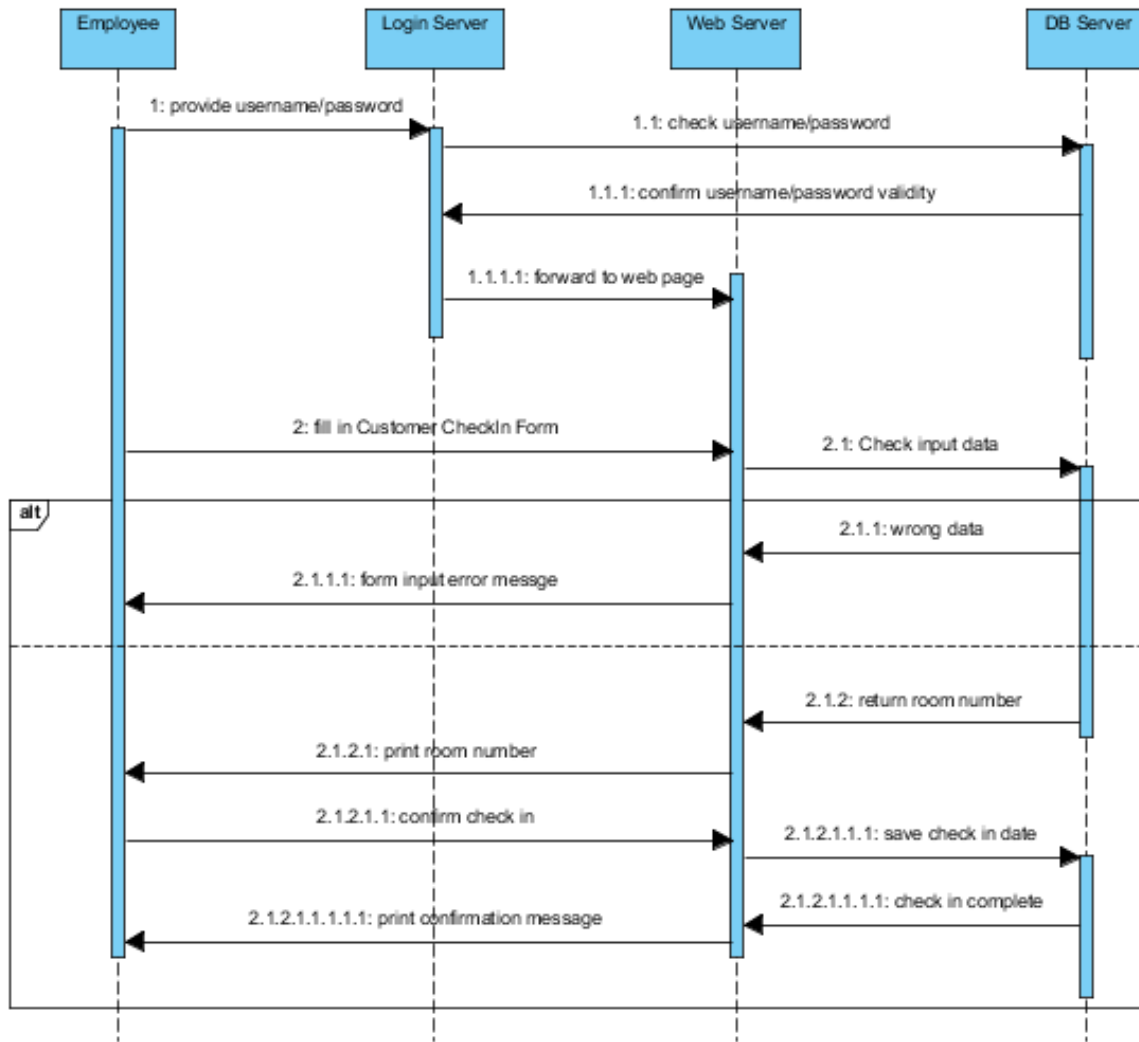
7 Διάγραμμα κλάσεων



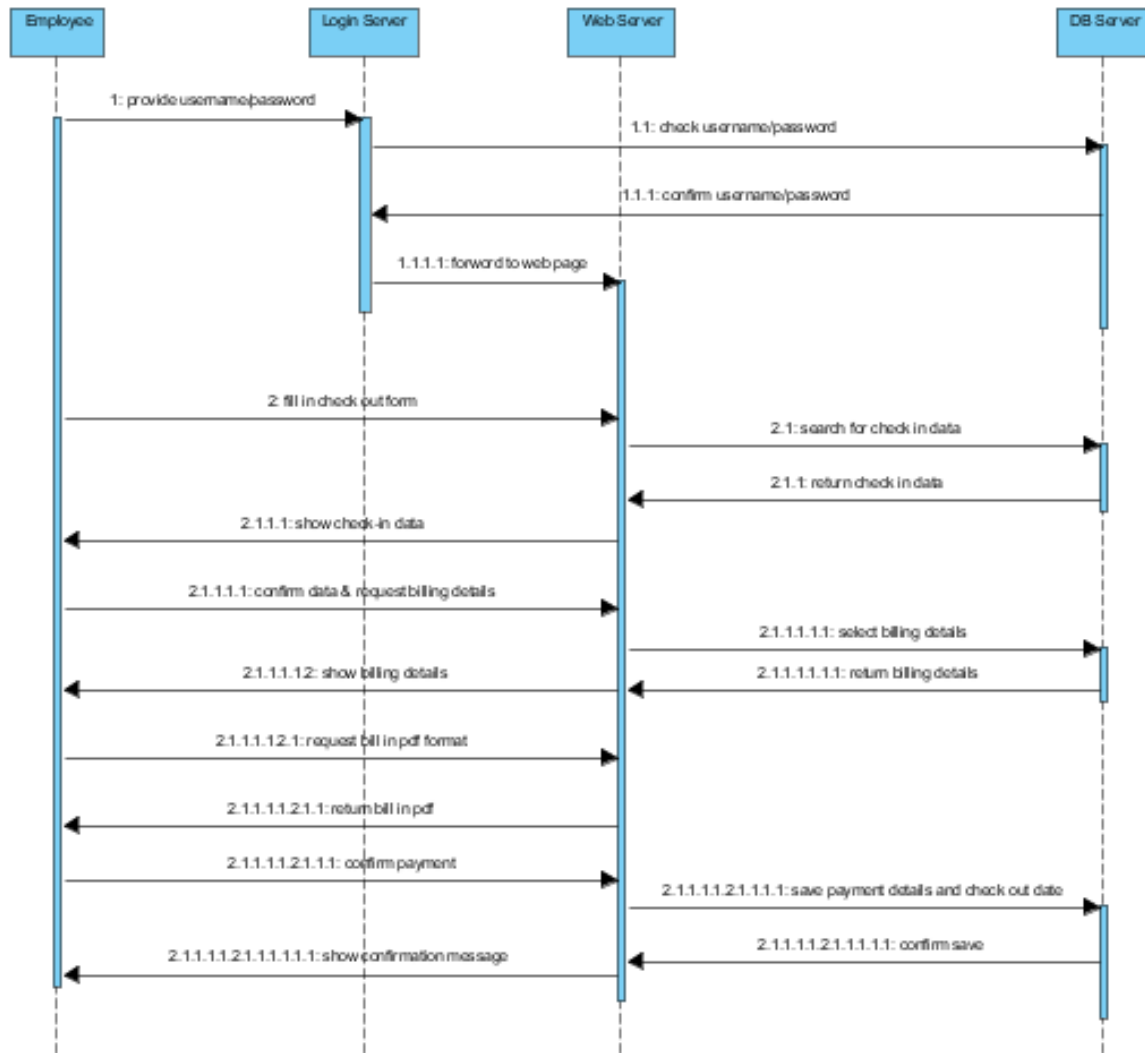
8 Διαγράμματα ακολουθίας



CheckInSeqDiagram

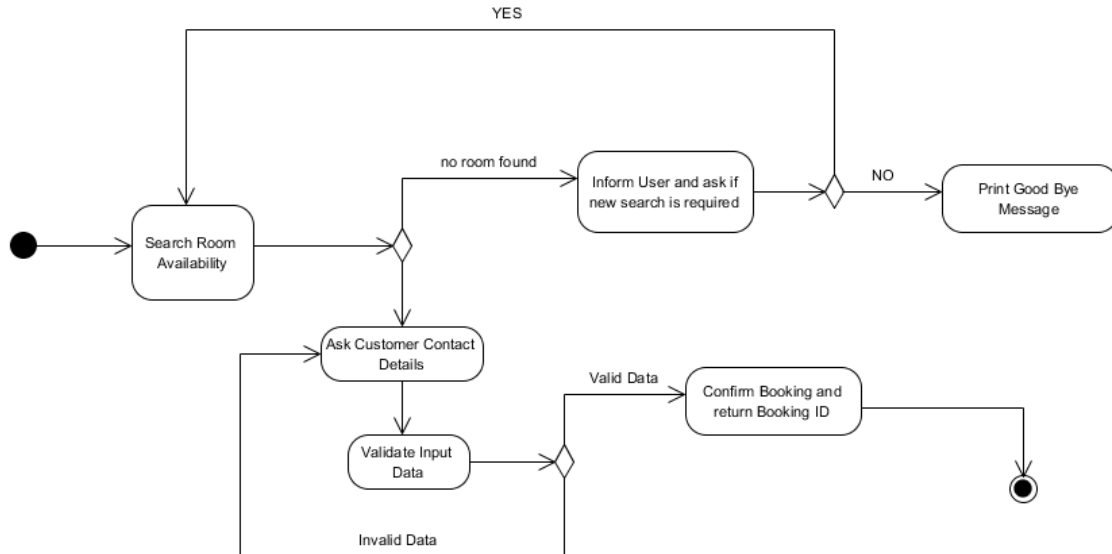


CheckOutSeqDiagram

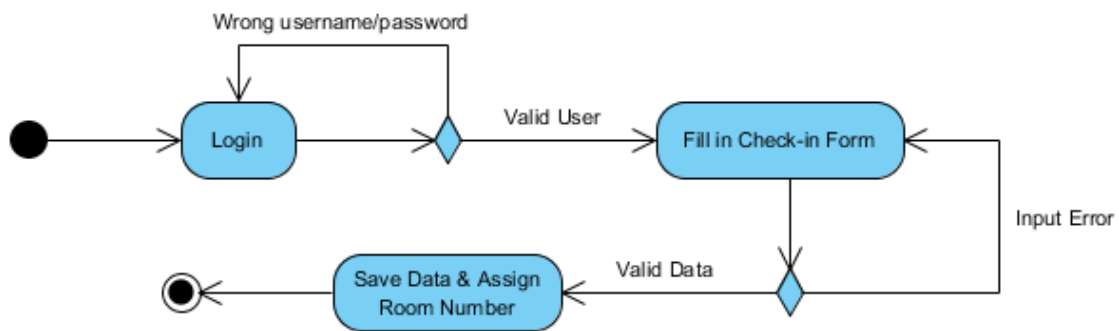


9 Διαγράμματα δραστηριοτήτων

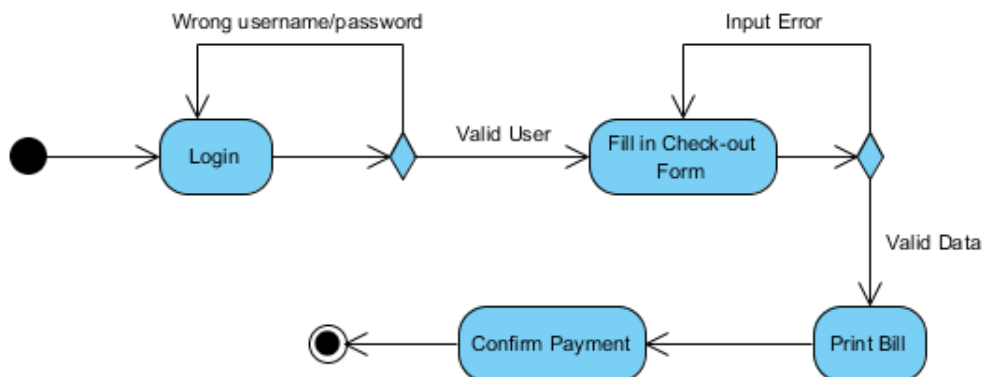
BookRoomActivity



CheckInActivity



CheckOutActivity



10 Αποτελέσματα static analysis PMD πριν και μετά

10.1 PMD πριν

Παρακαλώ ανατρέξτε στο αρχείο pmd-report_v1.csv που περιλαμβάνεται στο zip της εργασίας.

10.2 PMD μετά

Παρακαλώ ανατρέξτε στο αρχείο pmd-report_v2.csv που περιλαμβάνεται στο zip της εργασίας.

11 Μέθοδοι Unit esting

Όπως φαίνεται παρακάτω η μέθοδος unit testing που χρησιμοποιήσαμε επιτυγχάνει κάλυψη **29,7%**. Οι προδιαγραφές απαιτούσαν τουλάχιστον 25% κάλυψη. Αναλυτικότερα στοιχεία για την κάλυψη του κώδικα θα βρείτε εκτελώντας το αρχείο index.html που βρίσκεται στο φάκελο coverageHTML της εργασίας μας.

The screenshot displays an IDE interface with the following components:

- Top Bar:** Pack, JUnit, Navi, Outli.
- Left Panel:** Test results showing "Finished after 0.857 seconds", "Runs: 5/5", "Errors: 0", and "Failures: 0". It lists two test cases: "testcases.SQLManagerTest [Runner: JUnit 3] (0.834)" and "testcases.UserManagerTest [Runner: JUnit 3] (0.005)".
- Main Editor:** Displays the source code of "SQLManagerTest.java". The code includes assertions for adding charges and checking in/out details. A "private Customer getDummyCustomer()" method is also visible.
- Bottom Panel:** Contains a "Problems" tab showing "All Tests (0 errors, 271 warnings, 31 others)" and a "Coverage" tab showing a table of coverage data.

The Coverage tab table is as follows:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
onlineHote	29.7 %	1,377	3,257	4,634

SQLManagerTest.java Junit

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure with 'testcases' and 'SQLManagerTest'.
- JUnit View:** Displays test results: 'Runs: 3/3', 'Errors: 0', 'Failures: 0'. A green progress bar indicates successful execution.
- SQLManagerTest.java:** The main code file is open, showing the following code:

```
package testcases;

import java.util.Calendar;

public class SQLManagerTest extends TestCase {

    private SQLManager manager;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        manager = new SQLManager();
    }

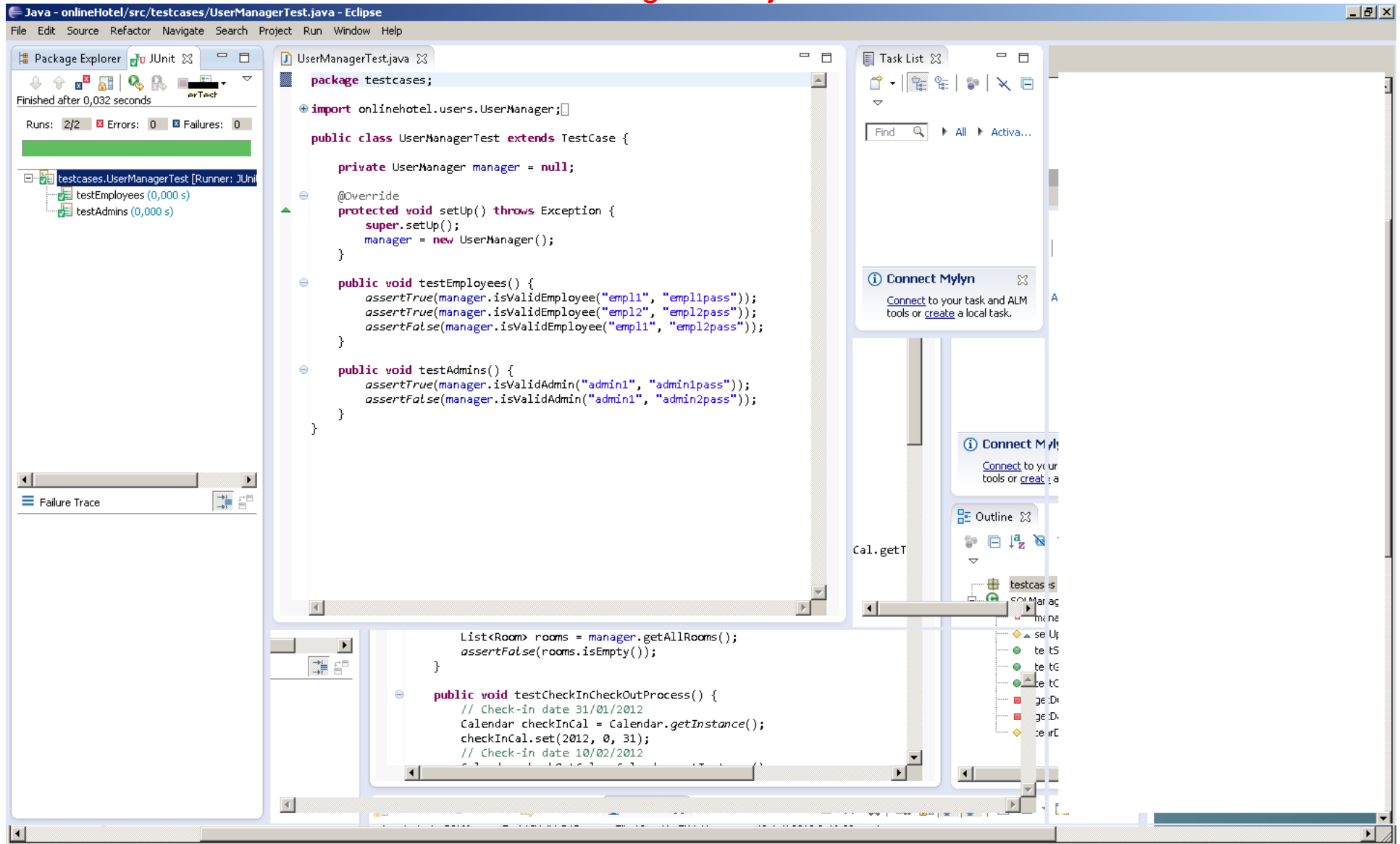
    public void testSearchAvailableRooms() {
        // Check-in date 31/01/2012
        Calendar checkInCal = Calendar.getInstance();
        checkInCal.set(2012, 0, 31);
        // Check-in date 10/02/2012
        Calendar checkOutCal = Calendar.getInstance();
        checkOutCal.set(2012, 1, 10);
        List<Room> rooms = manager.searchAvailableRooms(2, checkInCal.getTime(), checkOutCal.getTime());
        assertTrue(rooms.size() == 3);
    }

    public void testGetAllRooms(){
        List<Room> rooms = manager.getAllRooms();
        assertFalse(rooms.isEmpty());
    }

    public void testCheckInCheckOutProcess() {
        // Check-in date 31/01/2012
        Calendar checkInCal = Calendar.getInstance();
        checkInCal.set(2012, 0, 31);
        // Check-in date 10/02/2012
        Calendar checkOutCal = Calendar.getInstance();
        checkOutCal.set(2012, 1, 10);
        manager.checkInCheckOutProcess(2, checkInCal.getTime(), checkOutCal.getTime());
    }
}
```
- Task List:** Shows a 'Connect Mylyn' button and a search bar.
- Outline:** Displays the class structure with methods: 'setUp()', 'testSearchAvailableRooms()', 'testGetAllRooms()', 'testCheckInCheckOutProcess()', 'tearDown()'. The 'tearDown()' method is highlighted.
- Console:** Shows the output of the JUnit test run:

```
<terminated> SQLManagerTest [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (9 Feb 2013 9:46:03 p.m.)
Insert : insert into check_in_details(ROOM_NUMBER, CHECK_IN_TIME, CHECK_OUT_TIME, BOOKING_DETAILS_ID, ACTIVE)
ID : 6
```
- Failure Trace:** A tab is visible but empty.
- Welcome Page:** A 'Welcome to the Eclipse IDE for Java' page is visible in the background.

UserManagerTest.java Junit



SQLManagerTest.java

```
package testcases;

import java.util.Calendar;
import java.util.List;

import junit.framework.TestCase;
import onlinehotel.model.Charge;
import onlinehotel.model.CheckInDetails;
import onlinehotel.model.ContactDetails;
import onlinehotel.model.Customer;
import onlinehotel.model.Room;
import onlinehotel.model.db.SQLManager;

public class SQLManagerTest extends TestCase {

    private SQLManager manager;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        manager = new SQLManager();
    }

    public void testSearchAvailableRooms() {
        // Check-in date 31/01/2012
        Calendar checkInCal = Calendar.getInstance();
        checkInCal.set(2012, 0, 31);
        // Check-in date 10/02/2012
        Calendar checkOutCal = Calendar.getInstance();
        checkOutCal.set(2012, 1, 10);
        List<Room> rooms = manager.searchAvailableRooms(2, checkInCal.getTime(),
        checkOutCal.getTime());
        assertTrue(rooms.size() == 3);
    }

    public void testGetAllRooms(){
        List<Room> rooms = manager.getAllRooms();
        assertFalse(rooms.isEmpty());
    }

    public void testCheckInCheckOutProcess() {
        // Check-in date 31/01/2012
        Calendar checkInCal = Calendar.getInstance();
        checkInCal.set(2012, 0, 31);
        // Check-in date 10/02/2012
        Calendar checkOutCal = Calendar.getInstance();
        checkOutCal.set(2012, 1, 10);

        // 1 - Check Availability
        List<Room> rooms = manager.searchAvailableRooms(3, checkInCal.getTime(),
        checkOutCal.getTime());
        assertFalse(rooms.isEmpty());
        // 2 - Book the first room from the list
        Room room2Book = rooms.get(0);
        int bookingId = manager.addBookingForCustomer(getDummyCustomer(), room2Book,
        getDateAsString(1,31), getDateAsString(2,10));
    }
}
```

```

        assertTrue(bookingId >= 0);
        // 3 - Complete Check-in
        String roomNumber = manager.completeCheckIn(""+bookingId);
        assertNotNull(roomNumber);
        assertTrue(roomNumber.equals(room2Book.getRoomNumber()));
        // 4 - add additional expenses
        int chargeId = manager.addNewCharge(roomNumber, "e1", 10.0);
        assertTrue(chargeId >= 0);
        chargeId = manager.addNewCharge(roomNumber, "e2", 20.0);
        assertTrue(chargeId >= 0);
        chargeId = manager.addNewCharge(roomNumber, "e3", 34.0);
        assertTrue(chargeId >= 0);
        // 5 - get check-in details
        CheckInDetails details = manager.getCheckInDetails(roomNumber);
        assertNotNull(details);
        assertFalse(details.getAdditionalCharges().isEmpty());
        assertTrue(details.getAdditionalCharges().size() == 3);
        // 6 - complete check-out
        assertTrue(manager.completeCheckOut(details));
    }

    private Customer getDummyCustomer() {
        ContactDetails contactDetails = new ContactDetails();
        contactDetails.setAddress("TestAddress");
        contactDetails.setCellNumber("0000000000");
        contactDetails.setEmail("test@yahoo.gr");
        contactDetails.setPhoneNumber("1111111111");
        Customer customer = new Customer();
        customer.setFirstName("TestName");
        customer.setLastName("TestLastName");
        customer.setIdCardNumber("3456633");
        customer.setContactDetails(contactDetails);
        return customer;
    }

    private String getDateAsString(int month, int day) {
        return "2013-" + month + "-" + day;
    }

    @Override
    protected void tearDown() throws Exception {
        super.tearDown();
        manager.close();
    }
}

```

UserManagerTest.java

```
package testcases;

import onlinehotel.users.UserManager;
import junit.framework.TestCase;

public class UserManagerTest extends TestCase {

    private UserManager manager = null;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        manager = new UserManager();
    }

    public void testEmployees() {
        assertTrue(manager.isValidEmployee("empl1", "empl1pass"));
        assertTrue(manager.isValidEmployee("empl2", "empl2pass"));
        assertFalse(manager.isValidEmployee("empl1", "empl2pass"));
    }

    public void testAdmins() {
        assertTrue(manager.isValidAdmin("admin1", "admin1pass"));
        assertFalse(manager.isValidAdmin("admin1", "admin2pass"));
    }
}
```


12 Πηγαίος κώδικας – Screenshots – GUI

12.1 GUI CODE

GUI CheckInCheckOutPanel.Java

```
package onlinehotel.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import javax.swing.JTextField;

import onlinehotel.model.Charge;
import onlinehotel.model.CheckInDetails;
import onlinehotel.model.db.SQLManager;

public class CheckInCheckOutPanel extends JPanel {

    private SQLManager sqlManager;

    private JTextField bookingIdField = new JTextField();
    private JTextField roomNumberField = new JTextField();

    public CheckInCheckOutPanel(SQLManager sqlManager) {
        super();
        this.sqlManager = sqlManager;
        // Panel
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        add(createCheckInPanel());
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(new JSeparator(JSeparator.HORIZONTAL));
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(createCheckOutPanel());
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(new JSeparator(JSeparator.HORIZONTAL));
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(Box.createRigidArea(new Dimension(10, 380)));
    }

    private JComponent createCheckInPanel() {
        // Components
        JLabel bookingIdLabel = new JLabel("Booking Id : ");
        JButton checkInButton = new JButton("Check-in");
        checkInButton.addActionListener(new ActionListener() {
```

```

        @Override
        public void actionPerformed(ActionEvent arg0) {
            completeCheckIn();
        }
    });
    // Panel
    JPanel checkInPanel = new JPanel();
    checkInPanel.setLayout(new BoxLayout(checkInPanel, BoxLayout.LINE_AXIS));
    checkInPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    checkInPanel.add(bookingIdLabel);
    checkInPanel.add(Box.createRigidArea(new Dimension(17, 0)));
    checkInPanel.add(bookingIdField);
    checkInPanel.add(Box.createRigidArea(new Dimension(10, 0)));
    checkInPanel.add(checkInButton);
    return checkInPanel;
}

private JComponent createCheckOutPanel() {
    // Components
    JLabel roomNumLabel = new JLabel("Room Num. : ");
    JButton checkOutButton = new JButton("Check-out");
    checkOutButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            completeCheckOut();
        }
    });
    // Panel
    JPanel checkOutPanel = new JPanel();
    checkOutPanel.setLayout(new BoxLayout(checkOutPanel, BoxLayout.LINE_AXIS));
    checkOutPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    checkOutPanel.add(roomNumLabel);
    checkOutPanel.add(Box.createRigidArea(new Dimension(17, 0)));
    checkOutPanel.add(roomNumberField);
    checkOutPanel.add(Box.createRigidArea(new Dimension(10, 0)));
    checkOutPanel.add(checkOutButton);
    return checkOutPanel;
}

private JPanel createFieldWithLabel(Component field, String label, int space) {
    JPanel tfPanel = new JPanel();
    tfPanel.setLayout(new BoxLayout(tfPanel, BoxLayout.LINE_AXIS));
    tfPanel.add(new JLabel(label));
    tfPanel.add(Box.createRigidArea(new Dimension(space, 0)));
    tfPanel.add(field);
    return tfPanel;
}

private void completeCheckIn() {
    String bookingid = bookingIdField.getText();
    String roomNum = sqlManager.completeCheckIn(bookingid);
    if(bookingid != null && !bookingid.trim().equals("")) {
        if(roomNum != null) {
            JOptionPane.showMessageDialog(this, "Room for booking " +
bookingid + " is room " + roomNum,
                "Check-in", JOptionPane.INFORMATION_MESSAGE);
            bookingIdField.setText(null);
        } else {
            JOptionPane.showMessageDialog(this, "Check-in cannot be
completed!",
                "Check-in Eroor", JOptionPane.ERROR_MESSAGE);

```

```

    }
    } else {
        JOptionPane.showMessageDialog(this, "Please give a valid booking id!",
            "Check-in Erroor", JOptionPane.ERROR_MESSAGE);
    }
}

private void completeCheckOut() {
    String roomNum = roomNumberField.getText();
    if (roomNum != null && !roomNum.trim().equals("")) {
        CheckInDetails details = sqlManager.getCheckInDetails(roomNum);
        if (details != null) {
            boolean ok = sqlManager.completeCheckOut(details);
            if (ok) {
                String bill = getBillAsStr(details);
                JOptionPane.showMessageDialog(this, "Check-out completed!\n
Bill : \n" + bill,
                    "Check-in", JOptionPane.INFORMATION_MESSAGE);
                roomNumberField.setText(null);
            } else {
                JOptionPane.showMessageDialog(this, "Cannot complete check-
out",
                    "Check-out Erroor", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this, "Cannot complete check-out",
                "Check-out Erroor", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Please give a valid room number!",
            "Check-out Erroor", JOptionPane.ERROR_MESSAGE);
    }
}

private String getBillAsStr(CheckInDetails details) {
    StringBuilder buffer = new StringBuilder();
    Double sum = 0.0;
    List<Charge> list = details.getAdditionalCharges();
    for (Charge c : list) {
        buffer.append(c).append("\n");
        sum += c.getPrice();
    }
    long d1 = details.getCheckOutTime().getTime();
    long d2 = details.getCheckInTime().getTime();
    long days = Math.abs((d1 - d2) / (1000 * 60 * 60 * 24));
    sum += days * details.getBookingDetails().getPriceWithDiscount();
    buffer.append("Total cost is : ").append(sum);
    return buffer.toString();
}
}

```

GUI LoginFrame.java

```
package onlinehotel.gui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

import onlinehotel.users.UserManager;

public class LoginFrame extends JFrame {

    private UserManager manager = new UserManager();

    // Components
    private JComboBox loginAsCombo = new JComboBox(new String[]
{"Customer", "Employee", "Admin"});
    private JTextField usernameField = new JTextField(20);
    private JPasswordField passwordField = new JPasswordField(20);

    public LoginFrame() {
        super();
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setLocation(400, 200);
        //Add content to the window.
        this.add(this.createLoginPanel(), BorderLayout.CENTER);
    }

    public void displayWindow() {
        this.pack();
        this.setVisible(true);
    }

    private JComponent createLoginPanel() {
        JPanel loginPanel = new JPanel();
        loginPanel.setLayout(new BoxLayout(loginPanel, BoxLayout.Y_AXIS));
        loginPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 0));

        loginPanel.add(this.createLoginAsComponent());
        loginPanel.add(Box.createRigidArea(new Dimension(0, 10)));
        loginPanel.add(this.createUsernameComponent());
        loginPanel.add(Box.createRigidArea(new Dimension(0, 10)));
    }
}
```

```

        loginPanel.add(this.createPasswordComponent());
        loginPanel.add(Box.createRigidArea(new Dimension(0, 10)));
        loginPanel.add(this.createButtonsComponent());
        return loginPanel;
    }

    private JComponent createLoginAsComponent() {
        // Components
        JLabel loginAsLabel = new JLabel("Login as : ");
        loginAsCombo.setSelectedIndex(0);
        loginAsCombo.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                JComboBox comboBox = (JComboBox)event.getSource();
                int value = comboBox.getSelectedIndex();
                if (value == 0) {
                    usernameField.setEnabled(false);
                    usernameField.setText(null);
                    passwordField.setEnabled(false);
                    passwordField.setText(null);
                } else {
                    usernameField.setEnabled(true);
                    passwordField.setEnabled(true);
                }
            }
        });
        // Panel
        JPanel loginAsPanel = new JPanel();
        loginAsPanel.setLayout(new BoxLayout(loginAsPanel, BoxLayout.LINE_AXIS));
        loginAsPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
        loginAsPanel.add(loginAsLabel);
        loginAsPanel.add(Box.createRigidArea(new Dimension(20, 0)));
        loginAsPanel.add(loginAsCombo);
        return loginAsPanel;
    }

    private JComponent createUsernameComponent() {
        // Components
        JLabel label = new JLabel("Username : ");
        usernameField.setEnabled(false);
        // Panel
        JPanel component = new JPanel();
        component.setLayout(new BoxLayout(component, BoxLayout.LINE_AXIS));
        component.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
        component.add(label);
        component.add(Box.createRigidArea(new Dimension(10, 0)));
        component.add(usernameField);
        return component;
    }

    private JComponent createPasswordComponent() {
        // Components
        JLabel label = new JLabel("Password : ");
        passwordField.setEnabled(false);
        // Panel
        JPanel component = new JPanel();
        component.setLayout(new BoxLayout(component, BoxLayout.LINE_AXIS));
        component.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
        component.add(label);
        component.add(Box.createRigidArea(new Dimension(12, 0)));
        component.add(passwordField);
    }

```

```

        return component;
    }

    private JComponent createButtonsComponent() {
        // Components
        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int loginAs = loginAsCombo.getSelectedIndex();
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                loginUser(username, password, loginAs);
            }
        });
        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent arg0) {
                System.out.println("Cancel, going to terminate!");
                System.exit(0);
            }
        });
        // Panel
        JPanel component = new JPanel();
        component.setLayout(new BoxLayout(component, BoxLayout.LINE_AXIS));
        component.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
        component.add(okButton);
        component.add(Box.createRigidArea(new Dimension(10, 0)));
        component.add(cancelButton);
        return component;
    }

    private void loginUser(String username, String password, final int mode) {
        boolean isValidUser = false;
        if(mode == 0) {
            isValidUser = true;
        } else if (mode == 1) {
            isValidUser = manager.isValidEmployee(username, password);
        } else {
            isValidUser = manager.isValidAdmin(username, password);
        }
        if(isValidUser) {
            System.out.println("User is valid");
            this.setVisible(false);
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    //Turn off metal's use of bold fonts
                    MainTabbedFrame mainFrame = new MainTabbedFrame(mode);
                }
            });
        } else {
            System.out.println("User is NOT valid");
            JOptionPane.showMessageDialog(this, "Wrong username/password",
                "Login error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

GUI MainTabbedFrame.java

```
package onlinehotel.gui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;

import onlinehotel.model.db.SQLManager;

public class MainTabbedFrame extends JFrame {

    private static final int CUSTOMER_MODE = 0;
    private static final int EMPLOYEE_MODE = 1;
    private static final int ADMIN_MODE = 2;

    private SQLManager sqlManager;

    public MainTabbedFrame(int mode) {
        super("Hotel Application");

        sqlManager = new SQLManager();

        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setLocation(320, 80);
        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosed(WindowEvent e) {
                super.windowClosed(e);
                sqlManager.close();
                System.exit(0);
            }
        });
        //Add content to the window.
        this.add(this.createMainTabbedPanel(mode), BorderLayout.CENTER);

        //Display the window.
        this.pack();
        this.setVisible(true);
    }

    private JPanel createMainTabbedPanel(int mode) {
        JPanel panel = new JPanel(new GridLayout(1, 1));
        JTabbedPane tabbedPane = new JTabbedPane();

        if(mode == CUSTOMER_MODE) {
            JComponent tab = new SearchBookRoomPanel(sqlManager);
            tabbedPane.addTab("Search/Book Room", null, tab, "Search for available rooms and book a selected room");
        } else if (mode == EMPLOYEE_MODE) {
            JComponent tab1 = new SearchBookRoomPanel(sqlManager);
```

```

        tabbedPane.addTab("Search/Book Room", null, tab1, "Search for available
rooms and book a selected room");
        JComponent tab2 = new CheckInCheckOutPanel(sqlManager);
        tabbedPane.addTab("Check-in/Check-out", null, tab2, "Check-in/Check-
out");

        JComponent tab3 = new ChargesPanel(sqlManager);
        tabbedPane.addTab("Charges", null, tab3, "Charges");
    } else {
        JComponent tab = new OfferPanel(sqlManager);
        tabbedPane.addTab("Offers Management", null, tab, "Offers Management");
    }

    // Add the tabbed pane to this panel.
    panel.add(tabbedPane);

    // The following line enables to use scrolling tabs.
    tabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL_TAB_LAYOUT);

    return panel;
}

protected JComponent makeTextPanel(String text) {
    JPanel panel = new JPanel(false);
    JLabel filler = new JLabel(text);
    filler.setHorizontalAlignment(JLabel.CENTER);
    panel.setLayout(new GridLayout(1, 1));
    panel.add(filler);
    return panel;
}
}

```


GUI OfferPanel.java

```
package onlinehotel.gui;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

import onlinehotel.model.Room;
import onlinehotel.model.db.SQLManager;

public class OfferPanel extends JPanel {

    private SQLManager sqlManager;

    private JComboBox roomSelection = new JComboBox();
    private JTextField discountField = new JTextField();

    public OfferPanel(SQLManager sqlManager) {
        super();
        this.sqlManager = sqlManager;
        // Panel
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 0));
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(createFieldWithLabel(roomSelection, "Room : ", 9));
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(createFieldWithLabel(discountField, "Discount (%) : ", 12));
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(createAddChargeButton());
        add(Box.createRigidArea(new Dimension(300, 200)));
        setUpComponents();
    }

    private void setUpComponents() {
        List<Room> rooms = sqlManager.getAllRooms();
        for (Room room : rooms) {
            roomSelection.addItem(room);
        }
    }

    private JPanel createFieldWithLabel(JComponent textField, String label, int space)
    {
        JPanel tfPanel = new JPanel();
        tfPanel.setLayout(new BoxLayout(tfPanel, BoxLayout.LINE_AXIS));
        tfPanel.add(new JLabel(label));
        tfPanel.add(Box.createRigidArea(new Dimension(space, 0)));
        tfPanel.add(textField);
    }
}
```

```

        tfPanel.add(Box.createRigidArea(new Dimension(10, 0)));
        return tfPanel;
    }

    private JButton createAddChargeButton() {
        JButton button = new JButton("Save Discount");
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveDiscount();
            }
        });
        return button;
    }

    private void saveDiscount() {
        Room room = (Room)roomSelection.getSelectedItemAt(0);
        String discountStr = discountField.getText();
        try {
            Double discount = Double.parseDouble(discountStr);
            boolean ok = sqlManager.updateRoomWithDiscount(room, discount);
            if (ok) {
                JOptionPane.showMessageDialog(this, "Room updated!",
                    "Process Complete", JOptionPane.INFORMATION_MESSAGE);
                discountField.setText(null);
            } else {
                JOptionPane.showMessageDialog(this, "Cannot update room with
discount!",
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please give a valid discount
percent!",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

GUI OnlineHotel.java

```
package onlinehotel.gui;

import javax.swing.SwingUtilities;
import javax.swing.UIManager;

public class OnlineHotel {

    /**
     * Create the GUI and show it.  For thread safety,
     * this method should be invoked from
     * the event dispatch thread.
     */
    private static void createAndShowGUI() {
        LoginFrame frame = new LoginFrame();
        frame.displayWindow();
    }

    public static void main(String[] args) {
        //Schedule a job for the event dispatch thread:
        //creating and showing this application's GUI.
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                //Turn off metal's use of bold fonts
                UIManager.put("swing.boldMetal", Boolean.FALSE);
                createAndShowGUI();
            }
        });
    }
}
```

GUI SearchBookRoomPanel.java

```
package onlinehotel.gui;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import javax.swing.JTextField;

import onlinehotel.model.ContactDetails;
import onlinehotel.model.Customer;
import onlinehotel.model.Room;
import onlinehotel.model.db.SQLManager;

public class SearchBookRoomPanel extends JPanel {

    private static String[] MONTHS_ARRAY = {
        "-- Month --",
        "January", "February", "March",
        "April", "May", "June",
        "July", "August", "September",
        "October", "November", "December"
    };

    private static int[] MAX_DAYS_PER_MONTH = {
        31, // "-- Month --"
        31, // "January"
        29, // "February"
        31, // "March"
        30, // "April"
        31, // "May"
        30, // "June"
        31, // "July"
        31, // "August"
        30, // "September"
        31, // "October"
        30, // "November"
        31, // "December"
    };

    // Check-in fields
    private JComboBox checkInDay = new JComboBox();
    private JComboBox checkInMonth = new JComboBox(MONTHS_ARRAY);
    // Check-out fields

    private JComboBox checkOutDay = new JComboBox(new String[] { "3", "4" });
```

```

private JComboBox checkOutMonth = new JComboBox(MONTHS_ARRAY);
// Number of beds
private JComboBox bedsNum = new JComboBox(new String[] {"-----", "2", "3", "4"});
// Booking
private JComboBox roomsList = new JComboBox();
private JButton bookButton = new JButton("Book");
// Customer details
private JTextField firstNameField = new JTextField(20);
private JTextField lastNameField = new JTextField(20);
private JTextField phoneNumberField = new JTextField(20);
private JTextField cellNumberField = new JTextField(20);
private JTextField idCardNumberField = new JTextField(20);
private JTextField emailField = new JTextField(20);
private JTextField addressField = new JTextField(20);

private SQLManager sqlManager;

public SearchBookRoomPanel(SQLManager sqlManager) {
    super();
    this.sqlManager = sqlManager;
    // Panel
    setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
    setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 0));
    add(this.createCheckInPanel());
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(this.createCheckOutPanel());
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(this.createNumOfBedsPanel());
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(this.createSearchResetButtonsPanel());
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(new JSeparator(JSeparator.HORIZONTAL));
    add(Box.createRigidArea(new Dimension(0, 10)));
    createBookRoomPanel();
    // Initially all fields related to booking must be deactivated
    this.resetBookPanel();
}

private JComponent createCheckInPanel() {
    // Components
    JLabel checkInLabel = new JLabel("Check-in date : ");
    updateDaysComboBox(checkInDay, 0);
    checkInMonth.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            int monthSelection =
                ((JComboBox)event.getSource()).getSelectedIndex();
            updateDaysComboBox(checkInDay, monthSelection);
        }
    });
    // Panel
    JPanel checkInPanel = new JPanel();
    checkInPanel.setLayout(new BorderLayout(checkInPanel, BorderLayout.LINE_AXIS));
    checkInPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    checkInPanel.add(checkInLabel);
    checkInPanel.add(Box.createRigidArea(new Dimension(17, 0)));
    checkInPanel.add(checkInMonth);
    checkInPanel.add(Box.createRigidArea(new Dimension(10, 0)));
    checkInPanel.add(checkInDay);
    return checkInPanel;
}

```

```

private void updateDaysComboBox(JComboBox combo, int monthSelection) {
    combo.removeAllItems();
    combo.addItem("-- Day --");
    int maxDays = MAX_DAYS_PER_MONTH[monthSelection];
    for(int i=1; i <= maxDays; i++) {
        combo.addItem(""+i);
    }
}

private JComponent createCheckOutPanel() {
    // Components
    JLabel checkOutLabel = new JLabel("Check-out date : ");
    updateDaysComboBox(checkOutDay, 0);
    checkOutMonth.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            int monthSelection =
((JComboBox)event.getSource()).getSelectedIndex();
            updateDaysComboBox(checkOutDay, monthSelection);
        }
    });
    //Panel
    JPanel checkOutPanel = new JPanel();
    checkOutPanel.setLayout(new BoxLayout(checkOutPanel, BoxLayout.LINE_AXIS));
    checkOutPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    checkOutPanel.add(checkOutLabel);
    checkOutPanel.add(Box.createRigidArea(new Dimension(10, 0)));
    checkOutPanel.add(checkOutMonth);
    checkOutPanel.add(Box.createRigidArea(new Dimension(10, 0)));
    checkOutPanel.add(checkOutDay);
    return checkOutPanel;
}

private JComponent createNumOfBedsPanel() {
    // Components
    JLabel bedsLabel = new JLabel("Num. of beds : ");
    // Panel
    JPanel bedsPanel = new JPanel();
    bedsPanel.setLayout(new BoxLayout(bedsPanel, BoxLayout.LINE_AXIS));
    bedsPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    bedsPanel.add(bedsLabel);
    bedsPanel.add(Box.createRigidArea(new Dimension(14, 0)));
    bedsPanel.add(bedsNum);
    bedsPanel.add(Box.createRigidArea(new Dimension(200, 0)));
    return bedsPanel;
}

private JComponent createSearchResetButtonsPanel() {
    // Buttons
    JButton searchButton = new JButton("Search");
    searchButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            searchDBForRooms();
        }
    });
    JButton resetButton = new JButton("Reset");
    resetButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

```

```

        checkInMonth.setSelectedIndex(0);
        checkOutMonth.setSelectedIndex(0);
        bedsNum.setSelectedIndex(0);
        resetBookPanel();
    }
});
// Panel
JPanel searchResetPanel = new JPanel();
searchResetPanel.setLayout(new BorderLayout(searchResetPanel,
BoxLayout.LINE_AXIS));
searchResetPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
searchResetPanel.add(searchButton);
searchResetPanel.add(Box.createRigidArea(new Dimension(14, 0)));
searchResetPanel.add(resetButton);
searchResetPanel.add(Box.createRigidArea(new Dimension(90, 0)));
return searchResetPanel;
}

private void createBookRoomPanel() {
    // Components
    JLabel bookRoomLabel = new JLabel("Select a room : ");
    bookButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            roomBookingAction();
        }
    });
    // Room List
    JPanel roomListPanel = new JPanel();
    roomListPanel.setLayout(new BorderLayout(roomListPanel, BoxLayout.LINE_AXIS));
    roomListPanel.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
    roomListPanel.add(bookRoomLabel);
    roomListPanel.add(Box.createRigidArea(new Dimension(14, 0)));
    roomListPanel.add(roomsList);
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(roomListPanel);
    // firstname
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(firstNameField, "Firstname : ", 15));
    // lastname
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(lastNameField, "Lastname : ", 15));
    // phoneNumber
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(phoneNumberField, "Phone : ", 35));
    // cellNumber
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(cellNumberField, "Cell Phone : ", 10));
    // idCardNumber
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(idCardNumberField, "ID card : ", 32));
    // e-mail
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(emailField, "e-mail : ", 36));
    // address
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(createTextFieldWithLabel(addressField, "Address : ", 25));
    // Book Button
    add(Box.createRigidArea(new Dimension(0, 10)));
    add(bookButton);
}

```

```

private JPanel createTextFieldWithLabel(JTextField textField, String label, int
space) {
    JPanel tfPanel = new JPanel();
    tfPanel.setLayout(new BoxLayout(tfPanel, BoxLayout.LINE_AXIS));
    tfPanel.add(new JLabel(label));
    tfPanel.add(Box.createRigidArea(new Dimension(space, 0)));
    tfPanel.add(textField);
    return tfPanel;
}

private void resetBookPanel() {
    roomsList.removeAllItems();
    roomsList.addItem("-----");
    roomsList.setEnabled(false);
    bookButton.setEnabled(false);
    //
    firstNameField.setEnabled(false);
    lastNameField.setEnabled(false);
    phoneNumberField.setEnabled(false);
    cellNumberField.setEnabled(false);
    idCardNumberField.setEnabled(false);
    emailField.setEnabled(false);
    addressField.setEnabled(false);
    //
    firstNameField.setText(null);
    lastNameField.setText(null);
    phoneNumberField.setText(null);
    cellNumberField.setText(null);
    idCardNumberField.setText(null);
    emailField.setText(null);
    addressField.setText(null);
    //
    checkInDay.setEnabled(true);
    checkInMonth.setEnabled(true);
    checkOutDay.setEnabled(true);
    checkOutMonth.setEnabled(true);
}

private void searchDBForRooms() {
    // Check input
    int checkInDayValue = checkInDay.getSelectedIndex();
    int checkInMonthValue = checkInMonth.getSelectedIndex();
    int checkOutDayValue = checkOutDay.getSelectedIndex();
    int checkOutMonthValue = checkOutMonth.getSelectedIndex();
    int bedsNumValue = bedsNum.getSelectedIndex() + 1;
    if (checkInDayValue == 0 || checkInMonthValue == 0 ||
        checkOutDayValue == 0 || checkOutMonthValue == 0 ||
bedsNumValue == 0 )
    {
        JOptionPane.showMessageDialog(this, "All fields must be filled!",
            "Search Error", JOptionPane.ERROR_MESSAGE);
    } else if (
        (checkInMonthValue > checkOutMonthValue) ||
        (checkInMonthValue == checkOutMonthValue && checkInDayValue >=
checkOutDayValue) )
    {
        JOptionPane.showMessageDialog(this, "Please correct check-in/check-out
dates!",
            "Search Error", JOptionPane.ERROR_MESSAGE);
    } else {

```



```

        // Search DB for available rooms and show results
        Date checkInDate = getDate(checkInMonthValue, checkInDayValue);
        Date checkOutDate = getDate(checkOutMonthValue, checkOutDayValue);
        List<Room> availableRooms =
sqlManager.searchAvailableRooms(bedsNumValue, checkInDate, checkOutDate);
        if (availableRooms != null && availableRooms.size() > 0) {
            this.setAvailableRooms(availableRooms);
        } else {
            JOptionPane.showMessageDialog(this, "No results found within your
search criteria",
                                     "", JOptionPane.INFORMATION_MESSAGE);
        }
    }

private void setAvailableRooms(List<Room> rooms) {
    bookButton.setEnabled(true);
    roomsList.removeAllItems();
    roomsList.addItem("-----");
    for (Room room : rooms) {
        roomsList.addItem(room);
    }
    roomsList.setEnabled(true);
    firstNameField.setEnabled(true);
    lastNameField.setEnabled(true);
    phoneNumberField.setEnabled(true);
    cellNumberField.setEnabled(true);
    idCardNumberField.setEnabled(true);
    emailField.setEnabled(true);
    addressField.setEnabled(true);
    //
    checkInDay.setEnabled(false);
    checkInMonth.setEnabled(false);
    checkOutDay.setEnabled(false);
    checkOutMonth.setEnabled(false);
}

private Date getDate(int month, int day) {
    Calendar cal = Calendar.getInstance();
    cal.set(2013, month - 1, day);
    return cal.getTime();
}

private void roomBookingAction() {
    if (roomsList.getSelectedIndex() == 0) {
        JOptionPane.showMessageDialog(this, "You must select a room!",
                                     "Field Error", JOptionPane.ERROR_MESSAGE);
    } else {
        String address = addressField.getText();
        String email = emailField.getText();
        String phone = phoneNumberField.getText();
        String cellPhone = cellNumberField.getText();
        String firstName = firstNameField.getText();
        String lastName = lastNameField.getText();
        String idNum = idCardNumberField.getText();
        if( address != null && !address.trim().equals("") &&
            email != null && !email.trim().equals("") &&
            phone != null && !phone.trim().equals("") &&
            cellPhone != null && !cellPhone.trim().equals("") &&
            firstName != null && !firstName.trim().equals("") &&
            lastName != null && !lastName.trim().equals("") &&

```

```

        idNum != null && !idNum.trim().equals(""))
    ) {
        Room selectedRoom = (Room)roomsList.getSelectedItemAt();
        // Contact Details
        ContactDetails cd = new ContactDetails();
        cd.setAddress(address);
        cd.setCellNumber(cellPhone);
        cd.setEmail(email);
        cd.setPhoneNumber(phone);
        // Customer
        Customer c = new Customer();
        c.setFirstName(firstNameField.getText());
        c.setLastName(lastNameField.getText());
        c.setIdCardNumber(idCardNumberField.getText());
        c.setContactDetails(cd);
        // Check-in, Check-out dates
        int checkInDayValue = checkInDay.getSelectedIndex();
        int checkInMonthValue = checkInMonth.getSelectedIndex();
        int checkOutDayValue = checkOutDay.getSelectedIndex();
        int checkOutMonthValue = checkOutMonth.getSelectedIndex();
        String checkInDate = getDateAsString(checkInMonthValue,
checkInDayValue);
        String checkOutDate = getDateAsString(checkOutMonthValue,
checkOutDayValue);
        int bookingId = sqlManager.addBookingForCustomer(c, selectedRoom,
checkInDate, checkOutDate);
        if (bookingId >= 0) {
            JOptionPane.showMessageDialog(this, "Your booking id is " +
bookingId,
                "Booking Complete", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Booking cannot be
complete!",
                "Booking Error", JOptionPane.ERROR_MESSAGE);
        }
        this.resetBookPanel();
    } else {
        JOptionPane.showMessageDialog(this, "Please fill in all fields!",
            "Field Error", JOptionPane.ERROR_MESSAGE);
    }
}

private String getDateAsString(int month, int day) {
    return "2013-" + month + "-" + day;
}
}

```

12.2 Model CODE

Model BookingDetails.java

```
package onlinehotel.model;

import java.util.Calendar;
import java.util.Date;

public class BookingDetails {

    private String roomNumber;
    private Double priceOnBooking;
    private Double discountOnBooking;
    private Customer customer;
    private Date checkInDate;
    private Date checkOutDate;
    private Integer bookingId;

    public String getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(String roomNumber) {
        this.roomNumber = roomNumber;
    }

    public Double getPriceOnBooking() {
        return priceOnBooking;
    }

    public void setPriceOnBooking(Double priceOnBooking) {
        this.priceOnBooking = priceOnBooking;
    }

    public Double getDiscountOnBooking() {
        return discountOnBooking;
    }

    public void setDiscountOnBooking(Double discountOnBooking) {
        this.discountOnBooking = discountOnBooking;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public Date getCheckInDate() {
        return checkInDate;
    }

    public void setCheckInDate(Date checkInDate) {
        this.checkInDate = checkInDate;
    }

    public Date getCheckOutDate() {
```

```

        return checkOutDate;
    }

    public void setCheckOutDate(Date checkOutDate) {
        this.checkOutDate = checkOutDate;
    }

    public Integer getBookingId() {
        return bookingId;
    }

    public void setBookingId(Integer bookingId) {
        this.bookingId = bookingId;
    }

    public String getCheckInDateStr() {
        if (checkInDate != null) {
            return this.getStrFromDate(checkInDate);
        }
        return "";
    }

    public String getCheckOutDateStr() {
        if (checkOutDate != null) {
            return this.getStrFromDate(checkOutDate);
        }
        return "";
    }

    private String getStrFromDate(Date date) {
        Calendar c = Calendar.getInstance();
        c.setTime(date);
        int month = c.get(Calendar.MONTH) + 1;
        return "2013-" + month + "-" + c.get(Calendar.DAY_OF_MONTH);
    }

    public Double getPriceWithDiscount() {
        if (discountOnBooking != null)
            return (priceOnBooking * (100 - discountOnBooking)) / 100;
        else
            return priceOnBooking;
    }
}

```

Model Charge.java

```
package onlinehotel.model;

import java.text.DecimalFormat;

public class Charge {

    private String descr;
    private Double price;

    public String getDescr() {
        return descr;
    }

    public void setDescr(String descr) {
        this.descr = descr;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public String toString() {
        DecimalFormat df = new DecimalFormat("#.##");
        return descr + ", Price : " + df.format(price);
    }
}
```

Model CheckInDetails.java

```
package onlinehotel.model;

import java.sql.Timestamp;
import java.util.Date;
import java.util.List;

public class CheckInDetails {

    private int id;
    private String roomNumber;
    private BookingDetails bookingDetails;
    private Customer customer;
    private Date checkInTime;
    private Date checkOutTime;
    private List<Charge> additionalCharges;

    public String getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(String roomNumber) {
        this.roomNumber = roomNumber;
    }

    public BookingDetails getBookingDetails() {
        return bookingDetails;
    }

    public void setBookingDetails(BookingDetails bookingDetails) {
        this.bookingDetails = bookingDetails;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public Date getCheckInTime() {
        return checkInTime;
    }

    public void setCheckInTime(Date checkInTime) {
        this.checkInTime = checkInTime;
    }

    public Date getCheckOutTime() {
        return checkOutTime;
    }

    public void setCheckOutTime(Date checkOutTime) {
        this.checkOutTime = checkOutTime;
    }

    public List<Charge> getAdditionalCharges() {
        return additionalCharges;
    }
}
```

```
}

public void setAdditionalCharges(List<Charge> additionalCharges) {
    this.additionalCharges = additionalCharges;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}
}
```

Model ContactDetails.java

```
package onlinehotel.model;

public class ContactDetails {

    private String phoneNumber;
    private String cellNumber;
    private String email;
    private String address;

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getCellNumber() {
        return cellNumber;
    }

    public void setCellNumber(String cellNumber) {
        this.cellNumber = cellNumber;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

}
```


Model Customer.java

```
package onlinehotel.model;

public class Customer {

    private String firstName;
    private String lastName;
    private ContactDetails contactDetails;
    private String idCardNumber;
    private Integer customerId;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public ContactDetails getContactDetails() {
        return contactDetails;
    }

    public void setContactDetails(ContactDetails contactDetails) {
        this.contactDetails = contactDetails;
    }

    public String getIdCardNumber() {
        return idCardNumber;
    }

    public void setIdCardNumber(String idCardNumber) {
        this.idCardNumber = idCardNumber;
    }

    public Integer getCustomerId() {
        return customerId;
    }

    public void setCustomerId(Integer customerId) {
        this.customerId = customerId;
    }

}
```

Model Room.java

```
package onlinehotel.model;

import java.text.DecimalFormat;

public class Room {

    private String roomNumber;
    private Integer numOfBeds;
    private Double price;
    private Double discount;
    private String descr;

    public String getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(String roomNumber) {
        this.roomNumber = roomNumber;
    }

    public Integer getNumOfBeds() {
        return numOfBeds;
    }

    public void setNumOfBeds(Integer numOfBeds) {
        this.numOfBeds = numOfBeds;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public Double getDiscount() {
        return discount;
    }

    public void setDiscount(Double discount) {
        this.discount = discount;
    }

    public String getDescr() {
        return descr;
    }

    public void setDescr(String descr) {
        this.descr = descr;
    }

    public Double getPriceWithDiscount() {
        if(discount != null)
            return (price * ( 100 - discount)) / 100;
        else
            return price;
    }
}
```

```
    }

    public String toString() {
        DecimalFormat df = new DecimalFormat("#.##");
        String s = this.descr + ", Price : " + df.format(this.getPriceWithDiscount())
+ " \u20AC";
        return s;
    }
}
```

Model db SQLManager.java

```
package onlinehotel.model.db;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import onlinehotel.model.BookingDetails;
import onlinehotel.model.Charge;
import onlinehotel.model.CheckInDetails;
import onlinehotel.model.ContactDetails;
import onlinehotel.model.Customer;
import onlinehotel.model.Room;

public class SQLManager {

    private Connection connection;

    public SQLManager() {
        connection = SQLUtilities.connect();
    }

    public void close() {
        SQLUtilities.disconnect(connection);
    }

    public List<Room> searchAvailableRooms(Integer bedsNum, Date checkIn, Date
checkOut) {
        List<Room> rooms = getAllRooms(bedsNum);
        List<Room> availableRooms = new ArrayList<Room>();
        for (Room r : rooms) {
            List<BookingDetails> bookingDetails =
getOrderedBookingDetailsForRoom(r.getRoomNumber());
            if (roomIsAvailable(checkIn, checkOut, bookingDetails)) {
                availableRooms.add(r);
            }
        }
        return availableRooms;
    }

    public List<Room> getAllRooms() {
        List<Room> rooms = new ArrayList<Room>();
        String query = "select ROOM_NUMBER, PRICE, DISCOUNT, DESCR, NUM_OF_BEDS from
room";

        ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
        if (resultSet != null) {
            try {
                while (resultSet.next()) {
                    String roomNumber = resultSet.getString("ROOM_NUMBER");
                    Double price = resultSet.getDouble("PRICE");
                    Double discount = resultSet.getDouble("DISCOUNT");
                    String descr = resultSet.getString("DESCR");
                    Integer numOfBeds = resultSet.getInt("NUM_OF_BEDS");
                    Room r = new Room();
                    r.setDescr(descr);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return rooms;
    }
}
```

```

        r.setPrice(price);
        r.setRoomNumber(roomNumber);
        r.setDiscount(discount);
        r.setNumOfBeds(numOfBeds);
        rooms.add(r);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}
return rooms;
}

private List<Room> getAllRooms(Integer bedsNum) {
    List<Room> rooms = new ArrayList<Room>();
    String query = "select ROOM_NUMBER, PRICE, DISCOUNT, DESCR, NUM_OF_BEDS from
room "
        + "where NUM_OF_BEDS = " + bedsNum;
    ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
    if (resultSet != null) {
        try {
            while (resultSet.next()) {
                String roomNumber = resultSet.getString("ROOM_NUMBER");
                Double price = resultSet.getDouble("PRICE");
                Double discount = resultSet.getDouble("DISCOUNT");
                String descr = resultSet.getString("DESCR");
                Integer numOfBeds = resultSet.getInt("NUM_OF_BEDS");
                Room r = new Room();
                r.setDescr(descr);
                r.setPrice(price);
                r.setRoomNumber(roomNumber);
                r.setDiscount(discount);
                r.setNumOfBeds(numOfBeds);
                rooms.add(r);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return rooms;
}

private List<BookingDetails> getOrderedBookingDetailsForRoom(String roomNumber) {
    List<BookingDetails> bookingDetails = new ArrayList<BookingDetails>();
    String query = "select BOOKING_ID, ROOM_NUMBER, CHECK_IN_DATE, "
        + "CHECK_OUT_DATE, PRICE_ON_BOOKING, DISCOUNT_ON_BOOKING "
        + "from booking_details where ROOM_NUMBER = "
        + roomNumber + " order by CHECK_IN_DATE";
    ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
    if (resultSet != null) {
        try {
            while (resultSet.next()) {
                Integer id = resultSet.getInt("BOOKING_ID");
                String rm = resultSet.getString("ROOM_NUMBER");
                Double price = resultSet.getDouble("PRICE_ON_BOOKING");
                Double discount =
resultSet.getDouble("DISCOUNT_ON_BOOKING");
                Date bookingCheckInDate =
resultSet.getDate("CHECK_IN_DATE");
                Date bookingCheckOutDate =
resultSet.getDate("CHECK_OUT_DATE");

```

```

        BookingDetails b = new BookingDetails();
        b.setBookingId(id);
        b.setRoomNumber(rm);
        b.setPriceOnBooking(price);
        b.setDiscountOnBooking(discount);
        b.setCheckInDate(bookingCheckInDate);
        b.setCheckOutDate(bookingCheckOutDate);
        bookingDetails.add(b);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}
return bookingDetails;
}

private boolean roomIsAvailable(Date checkIn, Date checkOut, List<BookingDetails>
bookingDetails) {
    if(bookingDetails.isEmpty()) {
        return true;
    } else {
        BookingDetails currentBooking = null;
        BookingDetails previousBooking = null;
        for (BookingDetails details : bookingDetails) {
            previousBooking = currentBooking;
            currentBooking = details;
            if(previousBooking == null) {
                if ( currentBooking.getCheckInDate().after(checkOut) )
                    return true;
            } else {
                if (currentBooking.getCheckInDate().after(checkOut) &&
previousBooking.getCheckOutDate().before(checkIn) )
                    return true;
            }
        }
        // In case there is only one
        if ( currentBooking.getCheckOutDate().before(checkIn) )
            return true;
        return false;
    }
}

public int addBookingForCustomer(Customer c, Room r, String checkIn, String
checkOut) {
    int bookingID = -1;
    try {
        // Save contact details
        int cdID = saveContactDetails(c.getContactDetails());
        // Save Customer
        int cusID = saveCustomer(c, cdID);
        // Save booking
        bookingID = saveBooking(r, checkIn, checkOut, cusID);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return bookingID;
}

private int saveContactDetails(ContactDetails cd) throws SQLException {

```

```

        String query = "insert into contact_details(PHONE_NUMBER, CELL_PHONE,
ADDRESS, EMAIL) "
            + "VALUES('" + cd.getPhoneNumber() + "','" +
            + cd.getCellNumber() + "','" + cd.getAddress()
            + "','" + cd.getEmail() + "');"
        return SQLUtilities.executeInsert(query, connection);
    }

    private int saveCustomer(Customer c, int cdID) throws SQLException {
        String query = "insert into customer(FIRST_NAME, LAST_NAME,
CONTACT_DETAILS_ID, ID_CARD_NUMBER) "
            + "VALUES('" + c.getFirstName() + "','" +
            + c.getLastName() + "','" + cdID
            + "','" + c.getIdCardNumber() + "');"
        return SQLUtilities.executeInsert(query, connection);
    }

    private int saveBooking(Room r, String checkIn, String checkOut, int cusID) throws
SQLException {
        String query = "insert into booking_details(ROOM_NUMBER, CHECK_IN_DATE,
CHECK_OUT_DATE, CUSTOMER_ID, "
            + "PRICE_ON_BOOKING, DISCOUNT_ON_BOOKING) "
            + "VALUES('" + r.getRoomNumber() + "','" +
            + checkIn + "','" + checkOut
            + "','" + cusID + "','" + r.getPrice() + "','" +
r.getDiscount() + "');"
        return SQLUtilities.executeInsert(query, connection);
    }

    public String completeCheckIn(String bookingId) {
        BookingDetails b = getBookingDetailsById(bookingId);
        if (b != null) {
            int id = addCheckInDetailsRecord(b);
            System.out.println("ID : " + id);
            if (id > 0) {
                return b.getRoomNumber();
            }
        }
        return null;
    }

    private int addCheckInDetailsRecord(BookingDetails bd) {
        String query = "insert into check_in_details(ROOM_NUMBER, CHECK_IN_TIME,
CHECK_OUT_TIME, "
            + "BOOKING_DETAILS_ID, ACTIVE) "
            + "VALUES('" + bd.getRoomNumber() + "','" +
            + bd.getCheckInDateStr() + "','" + bd.getCheckOutDateStr()
            + "','" + bd.getBookingId() + "','" + 'YES' + "');"
        System.out.println("Insert : " + query);
        return SQLUtilities.executeInsert(query, connection);
    }

    private BookingDetails getBookingDetailsById(String id) {
        BookingDetails bookingDetails = null;
        String query = "select BOOKING_ID, ROOM_NUMBER, CHECK_IN_DATE, "
            + "CHECK_OUT_DATE, PRICE_ON_BOOKING, DISCOUNT_ON_BOOKING "
            + "from booking_details where BOOKING_ID = " + id;
        ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
        if (resultSet != null) {
            try {
                if (resultSet.next()) {

```

```

        String rm = resultSet.getString("ROOM_NUMBER");
        Double price = resultSet.getDouble("PRICE_ON_BOOKING");
        Double discount =
resultSet.getDouble("DISCOUNT_ON_BOOKING");
        Date bookingCheckInDate =
resultSet.getDate("CHECK_IN_DATE");
        Date bookingCheckOutDate =
resultSet.getDate("CHECK_OUT_DATE");

        bookingDetails = new BookingDetails();
        try {
            bookingDetails.setBookingId(Integer.parseInt(id));
        } catch (NumberFormatException e) {
            e.printStackTrace();
        }
        bookingDetails.setRoomNumber(rm);
        bookingDetails.setPriceOnBooking(price);
        bookingDetails.setDiscountOnBooking(discount);
        bookingDetails.setCheckInDate(bookingCheckInDate);
        bookingDetails.setCheckOutDate(bookingCheckOutDate);
    }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
return bookingDetails;
}

public CheckInDetails getCheckInDetails(String roomNumber) {
    CheckInDetails details = null;
    String query = "select ID, ROOM_NUMBER, CHECK_IN_TIME, CHECK_OUT_TIME, "
        + "BOOKING_DETAILS_ID "
        + "from check_in_details where ACTIVE = 'YES' AND ROOM_NUMBER = "
+ roomNumber;
    ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
    if (resultSet != null) {
        try {
            if (resultSet.next()) {
                String rm = resultSet.getString("ROOM_NUMBER");
                Date bookingCheckInDate =
resultSet.getDate("CHECK_IN_TIME");
                Date bookingCheckOutDate =
resultSet.getDate("CHECK_OUT_TIME");
                int bookingId = resultSet.getInt("BOOKING_DETAILS_ID");
                int id = resultSet.getInt("ID");

                details = new CheckInDetails();
                try {
                    details.setBookingDetails(getBookingDetailsById(bookingId+""));
                } catch (NumberFormatException e) {
                    e.printStackTrace();
                }
                details.setRoomNumber(rm);
                details.setId(id);
                details.setCheckInTime(bookingCheckInDate);
                details.setCheckOutTime(bookingCheckOutDate);

                details.setAdditionalCharges(getAdditionalChargesForCheckIn(id));
            }
        } catch (SQLException e) {

```



```

        e.printStackTrace();
    }
}
return details;
}

private List<Charge> getAdditionalChargesForCheckIn(int checkInId) {
    List<Charge> list = new ArrayList<Charge>();
    String query = "select DESCR, PRICE from charge where CHECK_IN_DETAILS_ID = "
+ checkInId;
    ResultSet resultSet = SQLUtilities.executeSQL(query, connection);
    if (resultSet != null) {
        try {
            while (resultSet.next()) {
                String descr = resultSet.getString("DESCR");
                Double price = resultSet.getDouble("PRICE");

                Charge c = new Charge();
                c.setPrice(price);
                c.setDescr(descr);
                list.add(c);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return list;
}

public int addNewCharge(String roomNumber, String descr, Double price) {
    CheckInDetails details = getCheckInDetails(roomNumber);
    if(details != null) {
        DecimalFormat df = new DecimalFormat("#.##");
        String query = "insert into charge(DESCR, PRICE, CHECK_IN_DETAILS_ID) "
+ "VALUES('" + descr + "', '"
+ df.format(price) + "', '" + details.getId() + "')";
        return SQLUtilities.executeInsert(query, connection);
    } else {
        return -1;
    }
}

public boolean completeCheckOut(CheckInDetails details) {
    if(details != null) {
        String query = "update check_in_details set ACTIVE = 'NO' where ID = "
+ details.getId();
        return SQLUtilities.executeUpdate(query, connection);
    }
    return false;
}

public boolean updateRoomWithDiscount(Room room, Double discount) {
    if(room != null && discount != null) {
        DecimalFormat df = new DecimalFormat("#.##");
        String query = "update room set DISCOUNT = '" + df.format(discount)
+ "' where ROOM_NUMBER = " + room.getRoomNumber();
        return SQLUtilities.executeUpdate(query, connection);
    }
    return false;
}
}

```

Model db SQLUtilities.java

```
package onlinehotel.model.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class SQLUtilities {

    public static Connection connect() {
        Connection connection = null;
        try {
            // Load the JDBC driver
            String driverName = "com.mysql.jdbc.Driver"; // MySQL MM JDBC driver
            Class.forName(driverName);
            // Create a connection to the database
            String url =
"jdbc:mysql://localhost:3308/hoteldb?useUnicode=true&characterEncoding=utf8";
            String username = "onlineHotel";
            String password = "onlineHotel";
            connection = DriverManager.getConnection(url, username, password);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

    public static ResultSet executeSQL(String query, Connection connection) {
        if (connection != null) {
            try {
                Statement statement = connection.createStatement();
                statement.execute(query);
                return statement.getResultSet();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        return null;
    }

    public static int executeInsert(String query, Connection connection) {
        if (connection != null) {
            try {
                Statement statement = connection.createStatement();
                statement.executeUpdate(query, Statement.RETURN_GENERATED_KEYS);
                ResultSet generatedKeys = statement.getGeneratedKeys();
                if (generatedKeys.next()) {
                    return generatedKeys.getInt(1);
                } else {
                    return -1;
                }
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

```

        }
    }
    return -1;
}

public static boolean executeUpdate(String query, Connection connection) {
    if (connection != null) {
        try {
            Statement statement = connection.createStatement();
            int num = statement.executeUpdate(query);
            if (num > 0) {
                return true;
            } else {
                return false;
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return false;
}

public static void disconnect(Connection connection) {
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

}

```

Users admins.properties

admin1=admin1pass

Users employees.properties

empl1=empl1pass

empl2=empl2pass

Users UserManager.java

```
package onlinehotel.users;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class UserManager {

    private Properties employees;
    private Properties admins;

    public UserManager() {
        employees = new Properties();
        admins = new Properties();
        try{
            InputStream in =
UserManager.class.getResourceAsStream("employees.properties");
            employees.load(in);
            in.close();

            in = UserManager.class.getResourceAsStream("admins.properties");
            admins.load(in);
            in.close();
        } catch (IOException ex) {
            System.err.println("Cannot load users!");
        }
    }

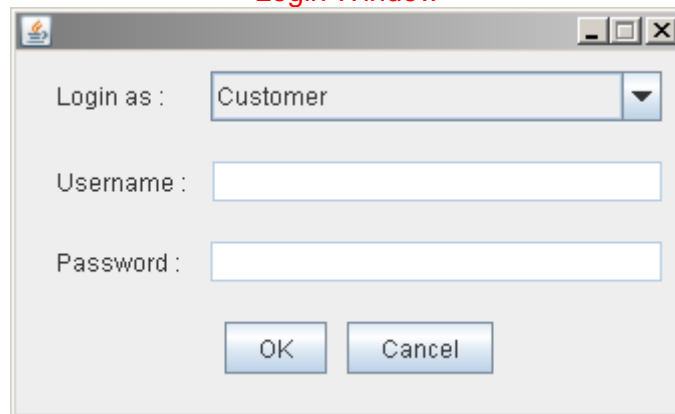
    public boolean isValidAdmin(String username, String password) {
        String realPassword = admins.getProperty(username);
        if (realPassword != null && realPassword.equals(password)) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isValidEmployee(String username, String password) {
        String realPassword = employees.getProperty(username);
        if (realPassword != null && realPassword.equals(password)) {
            return true;
        } else {
            return false;
        }
    }
}
```

12.3 Screenshots

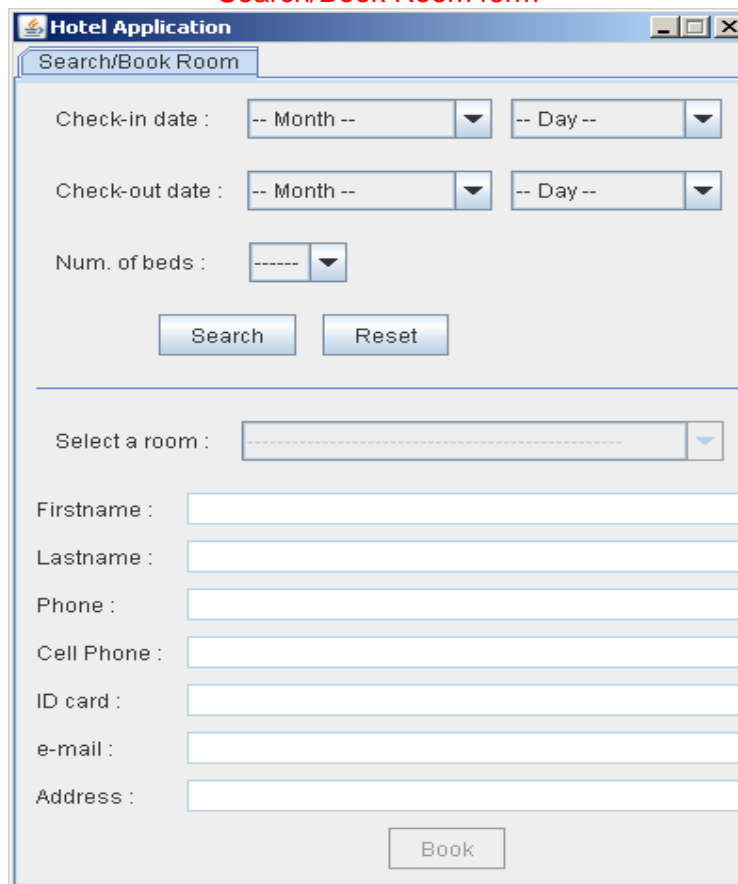
Customer Screenshots

Login Window



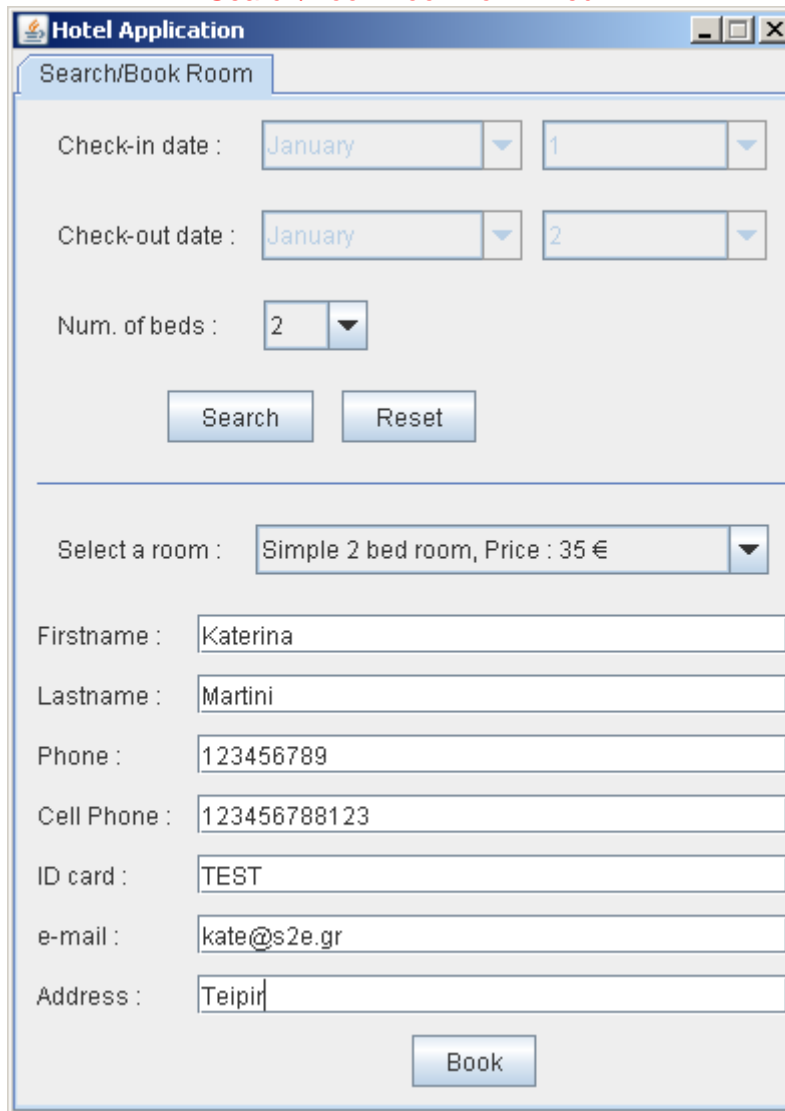
A screenshot of a login window titled "Login Window". It features a "Login as:" dropdown menu with "Customer" selected. Below this are two text input fields for "Username:" and "Password:". At the bottom are "OK" and "Cancel" buttons.

Search/Book Room form



A screenshot of the "Search/Book Room" form within the "Hotel Application". The form has a tabbed interface with the "Search/Book Room" tab selected. It includes fields for "Check-in date:" (Month and Day dropdowns), "Check-out date:" (Month and Day dropdowns), and "Num. of beds:" (a dropdown with "-----" selected). Below these are "Search" and "Reset" buttons. A horizontal line separates this section from the room selection section, which includes a "Select a room:" dropdown. Below the room selection are several text input fields for "Firstname:", "Lastname:", "Phone:", "Cell Phone:", "ID card:", "e-mail:", and "Address:". A "Book" button is located at the bottom right of the form.

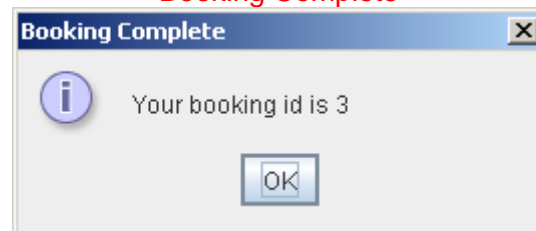
Search/Book Room form filled



The screenshot shows a web application window titled "Hotel Application" with a tab labeled "Search/Book Room". The form contains the following fields and controls:

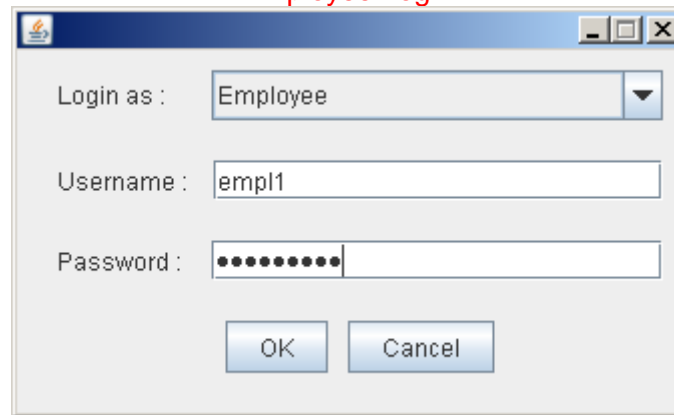
- Check-in date :** A dropdown menu showing "January" and a text box containing "1".
- Check-out date :** A dropdown menu showing "January" and a text box containing "2".
- Num. of beds :** A dropdown menu showing "2".
- Buttons:** "Search" and "Reset" buttons.
- Select a room :** A dropdown menu showing "Simple 2 bed room, Price : 35 €".
- Firstname :** A text box containing "Katerina".
- Lastname :** A text box containing "Martini".
- Phone :** A text box containing "123456789".
- Cell Phone :** A text box containing "123456788123".
- ID card :** A text box containing "TEST".
- e-mail :** A text box containing "kate@s2e.gr".
- Address :** A text box containing "Teipir".
- Book Button:** A "Book" button at the bottom right.

Booking Complete



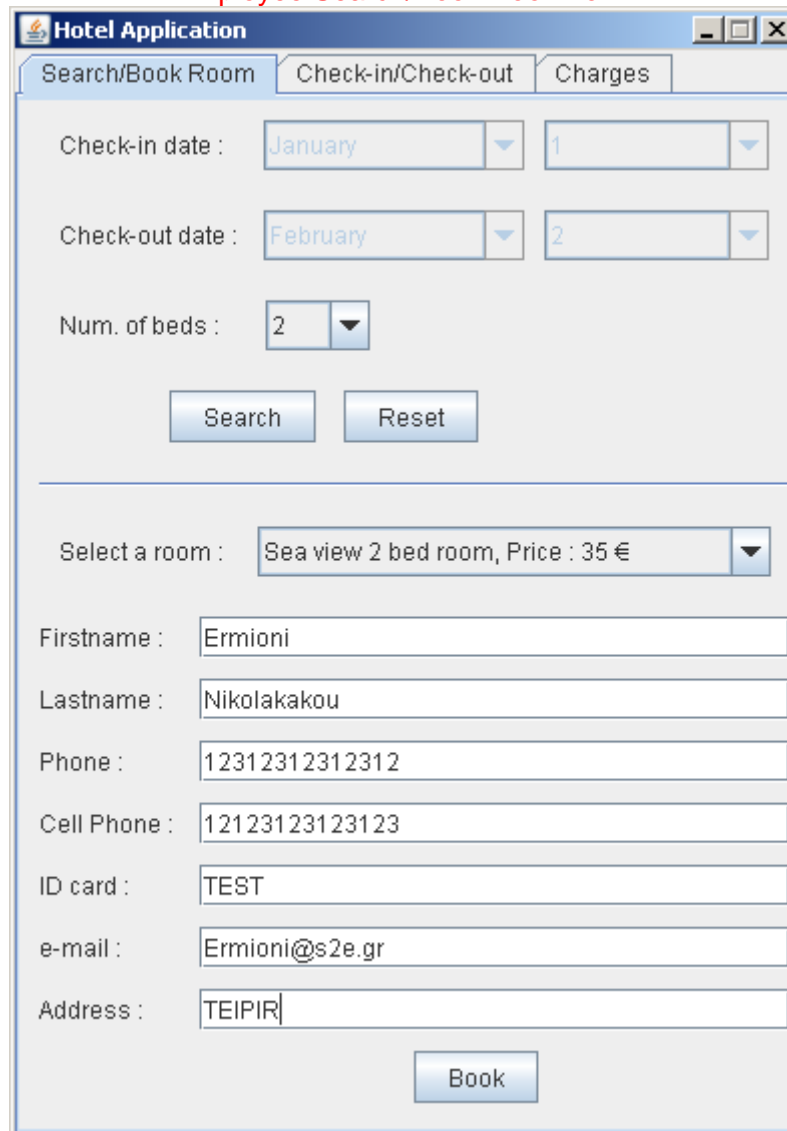
The screenshot shows a small dialog box titled "Booking Complete" with a close button (X) in the top right corner. It contains an information icon (i) and the text "Your booking id is 3". At the bottom, there is an "OK" button.

Employee Login



A dialog box titled "Employee Login" with a standard Windows window border. It contains three input fields: "Login as :" with a dropdown menu showing "Employee", "Username :" with a text box containing "empl1", and "Password :" with a text box containing masked characters (dots). Below the fields are two buttons: "OK" and "Cancel".

Employee Search/Book Room form



A form titled "Hotel Application" with a tabbed interface. The "Search/Book Room" tab is selected. The form contains several input fields and buttons. The "Check-in date" field has a dropdown for the month (January) and a dropdown for the day (1). The "Check-out date" field has a dropdown for the month (February) and a dropdown for the day (2). The "Num. of beds" field has a dropdown menu showing 2. Below these fields are "Search" and "Reset" buttons. A "Select a room" dropdown menu shows "Sea view 2 bed room, Price : 35 €". Below this are several text input fields for "Firstname" (Ermioni), "Lastname" (Nikolakakou), "Phone" (12312312312312), "Cell Phone" (12123123123123), "ID card" (TEST), "e-mail" (Ermioni@s2e.gr), and "Address" (TEIPIR). A "Book" button is located at the bottom right.

Employee Check-in/Check-out form


Hotel Application

Search/Book Room Check-in/Check-out Charges

Booking Id :

Room Num. :

Check-in

 Room for booking 3 is room 201

Employee Charges form

Hotel Application

Search/Book Room Check-in/Check-out **Charges**


Room Num. : 201

Description : test

Price : 100

Add Charge

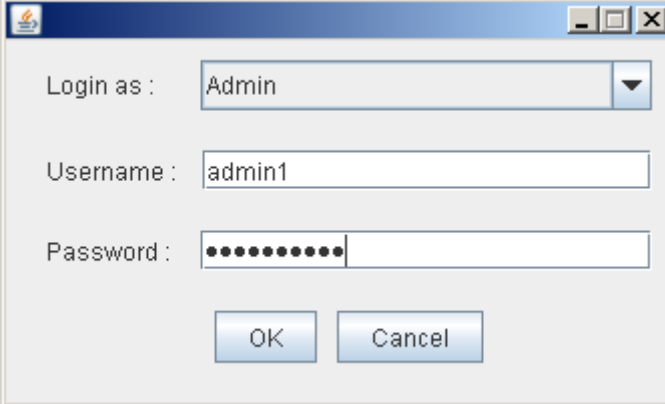
Process Complete

 Charge with description "test" and price 100 added for room 201

OK

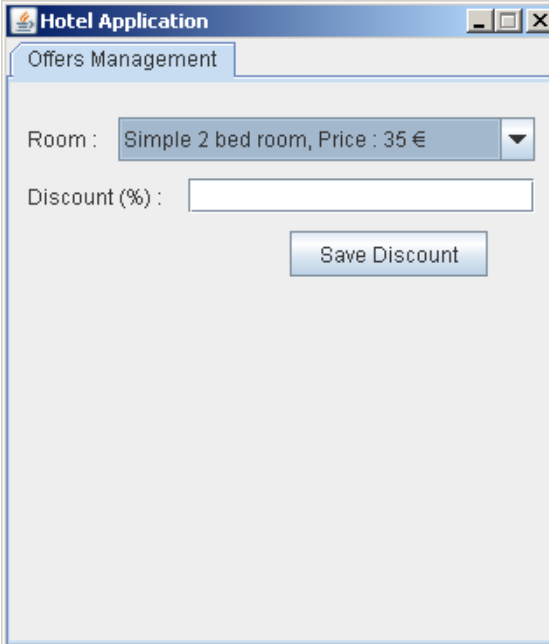
Admin Screenshots

Admin login window



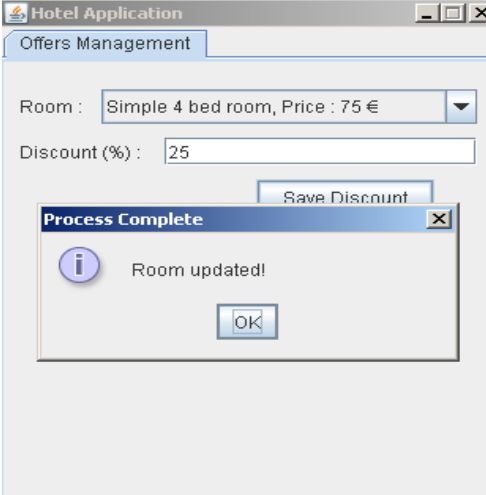
Admin login window showing fields for Login as (Admin), Username (admin1), Password (masked), and buttons for OK and Cancel.

Offers Management form



Offers Management form showing a Room dropdown menu (Simple 2 bed room, Price : 35 €), a Discount (%) input field, and a Save Discount button.

Room Discount Complete



Room Discount Complete form showing a Room dropdown menu (Simple 4 bed room, Price : 75 €), a Discount (%) input field (25), and a Save Discount button. A Process Complete dialog box is displayed, indicating Room updated!

Έλεγχος Αποδοχής Χρηστών (UAT)

13 User Acceptance Testing

Ορισμός Ελέγχου Αποδοχής χρηστών

- Ως acceptance testing ορίζουμε τα τεστ που τρέχει ο χρήστης ώστε να δει εάν η εφαρμογή πληροί τα κριτήρια που είχε θέσει στην αρχή της ανάπτυξης του λογισμικού.
- Πρόκειται για black box testing το οποίο δίνει την ευκαιρία στον πελάτη/project manager να επιβεβαιώσει ότι η λειτουργικότητα και η χρηστικότητα του προγράμματος βρίσκεται σε αποδεκτά σημεία προτού γίνει πλήρως αποδεκτό και μπει στην παραγωγή.
- Τα τεστ αποδοχής είναι ευθύνη των πελατών/project manager, όμως γίνονται σε στενή συνεργασία με την ομάδα προγραμματιστών που είναι υπεύθυνοι για το λογισμικό.

Υπεύθυνοι Ελέγχου Αποδοχής Χρηστών

Ρόλος	Όνομα	Καθήκοντα
Σπουδάστρια	Αικατερίνη Μαρτίνη	Σχεδιασμός και εκτέλεση ελέγχων αποδοχής χρηστών
Σπουδάστρια	Ερμιόνη Νικολακάκου	Σχεδιασμός και εκτέλεση ελέγχων αποδοχής χρηστών
Εργαστηριακός Συνεργάτης	Ιορδανάκης Μιχαήλ	Υπεύθυνος διαμόρφωσης κριτηρίων αποδοχής
Καθηγητής	Πρεζεράκος Γεώργιος	Υπεύθυνος συνολικής αποδοχής του έργου

Έλεγχος

1.	Test Case Customer forms (Search/Book rooms)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
1.1	Εκτέλεση εφαρμογής	Εμφάνιση παραθύρου login		Pass
1.2	Login σαν customer χωρίς password	Εμφάνιση φόρμας αναζήτησης δωματίων		Pass
1.3	Πάτημα πλήκτρου Search στη φόρμα αναζήτησης δωματίων χωρίς να συμπληρωθεί κανένα πεδίο	Εμφάνιση σφάλματος σχετικά με το ότι όλα τα πεδία θα πρέπει να συμπληρωθούν		Pass
1.4	Συμπλήρωση όλων των πεδίων της φόρμας αναζήτησης δωματίων και πάτημα του κουμπιού reset	Καθαρισμός όλων των πεδίων		Pass
1.5	Στη φόρμα αναζήτησης δωματίων επιλέγουμε checkin date που είναι μετά το check out date και πατάμε search	Εμφάνιση σφάλματος σχετικά με τις ημερομηνίες checkin και check out		Pass
1.6	Πάτημα πλήκτρου Book στη φόρμα αναζήτησης δωματίων χωρίς να συμπληρωθεί email και address	Εμφάνιση σφάλματος σχετικά με το ότι θα πρέπει να συμπληρωθούν όλα τα πεδία		Pass

1.7	Συμπλήρωση email πεδίου με απλό κείμενο που δεν είναι της μορφής something@domain.xx	Εμφάνιση σφάλματος σχετικά με το ότι δεν είναι έγκυρη μορφή email	Η καταχώρηση έγινε χωρίς να εμφανιστεί σφάλμα.	Fail
1.8	Αφού πατηθεί το κουμπί Book θα πρέπει να πηγαίνει mail επιβεβαίωσης της κράτησης στον πελάτη.	Αποστολή email επιβεβαίωσης κράτησης	To be implemented	Fail

2.	Test Case Login form			
Test #	Tasks	Expected Result	Notes	Pass /Fail
2.1	Login σαν Employee με λάθος password	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.2	Login σαν Employee με λάθος username	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.3	Επιλογή Login σαν Employee αλλά εισαγωγή των σωστών στοιχείων του admin	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.4	Επιλογή Login σαν Admin αλλά εισαγωγή των σωστών στοιχείων του employee	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.5	Επιλογή Login σαν Customer	Απενεργοποίηση πεδίων username και password		Pass
2.6	Στη φόρμα Login επιλέγουμε το πλήκτρο Cancel	Κλείσιμο εφαρμογής		Pass
2.7	Login σαν Admin με λάθος password	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.8	Login σαν Admin με λάθος username	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass
2.9	Επιλέγουμε Login σαν Employee και αφήνουμε κενά τα πεδία username, password και πατάμε OK	Εμφάνιση σφάλματος σχετικά με το Username/Password		
2.10	Επιλέγουμε Login σαν Admin και αφήνουμε κενά τα πεδία username, password και πατάμε OK	Εμφάνιση σφάλματος σχετικά με το Username/Password		Pass

3.	Test Case Employee forms (Search/Book room)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
3.1	Πάτημα πλήκτρου Search στη φόρμα αναζήτησης δωματίων χωρίς να συμπληρωθεί κανένα πεδίο	Εμφάνιση σφάλματος σχετικά με το ότι όλα τα πεδία θα πρέπει να συμπληρωθούν		Pass
3.2	Συμπλήρωση όλων των πεδίων της φόρμας αναζήτησης δωματίων και πάτημα του κουμπιού reset	Καθαρισμός όλων των πεδίων		Pass
3.3	Στη φόρμα αναζήτησης δωματίων επιλέγουμε checkin date που είναι μετά το check out date και πατάμε search	Εμφάνιση σφάλματος σχετικά με τις ημερομηνίες checkin και check out		Pass
3.4	Πάτημα πλήκτρου Book στη φόρμα αναζήτησης δωματίων χωρίς να συμπληρωθεί email και address	Εμφάνιση σφάλματος σχετικά με το ότι θα πρέπει να συμπληρωθούν όλα τα πεδία		Pass
3.5	Συμπλήρωση email πεδίου με απλό κείμενο που δεν είναι της μορφής something@domain.xx	Εμφάνιση σφάλματος σχετικά με το ότι δεν είναι έγκυρη μορφή email	Η καταχώρηση έγινε χωρίς να εμφανιστεί σφάλμα.	Fail
3.6	Δοκιμή λειτουργίας copy/paste σε όλα τα πεδία με ποντίκι	Αναμενόμενη λειτουργία copy/paste		Pass
3.7	Δοκιμή λειτουργίας copy/paste σε όλα τα πεδία με πληκτρολόγιο ctrl+c ctrl+v	Αναμενόμενη λειτουργία copy/paste		Pass
3.8				

4.	Test Case Employee forms (Check-in/Check-out)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
4.1	Πάτημα του κουμπιού Check-in χωρίς να πληκτρολογηθεί κάποια τιμή	Σφάλμα σχετικά με το Booking id		Pass
4.2	Πάτημα του κουμπιού Check-in πληκτρολογώντας μια τιμή με σύμβολα	Σφάλμα σχετικά με το Booking id		Pass
4.3	Εισαγωγή ενός έγκυρου Booking id και πάτημα του κουμπιού Check-in	Εμφάνιση μηνύματος με τον αριθμό δωματίου που αντιστοιχεί στο booking id που πληκτρολογήθηκε		Pass
4.4	Πάτημα του κουμπιού Check-out χωρίς να πληκτρολογηθεί κάποια τιμή	Σφάλμα σχετικά με το Room num		Pass
4.5	Πάτημα του κουμπιού Check-out πληκτρολογώντας μια τιμή με σύμβολα	Σφάλμα σχετικά με το Room num		Pass

4.6	Εισαγωγή ενός έγκυρου Room num και πάτημα του κουμπιού Check-out	Μήνυμα Check out completed / Απελευθέρωση δωματίου / Εμφάνιση λογαριασμού		Pass
-----	--	---	--	------

5.	Test Case Employee forms (Charges)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
5.1	Πάτημα του κουμπιού Charge χωρίς να συμπληρωθούν τα πεδία	Εμφάνιση μηνύματος σφάλματος σχετικά με τη μη συμπλήρωση πεδίων		Pass
5.2	Συμπλήρωση μη έγκυρου αριθμού δωματίου, αλλά έγκυρης περιγραφής και έγκυρης τιμής.	Εμφάνιση μηνύματος σφάλματος σχετικά με τη μη έγκυρη εγγραφή αριθμού δωματίου		Pass
5.3	Συμπλήρωση έγκυρου αριθμού δωματίου και έγκυρης τιμής αλλά αφήνουμε την περιγραφή κενή	Εμφάνιση μηνύματος σφάλματος σχετικά με μη συμπλήρωση της περιγραφής		Pass
5.4	Συμπλήρωση έγκυρου αριθμού δωματίου, έγκυρης περιγραφής. Στο πεδίο της τιμής εισάγουμε σύμβολα αντί για αριθμούς	Εμφάνιση μηνύματος σφάλματος σχετικά με μη έγκυρη τιμή στο πεδίο της τιμής.		Pass
5.5	Συμπλήρωση έγκυρου αριθμού δωματίου, έγκυρης περιγραφής. Στο πεδίο της τιμής εισάγουμε γράμματα αντί για αριθμούς.	Εμφάνιση σφάλματος σχετικά με μη έγκυρη τιμή στο πεδίο της τιμής.		Pass
5.6	Συμπλήρωση έγκυρου αριθμού δωματίου, έγκυρης περιγραφής και έγκυρης χρέωσης και πάτημα του κουμπιού Add Charge	Εμφάνιση μηνύματος επιτυχούς χρέωσης δωματίου. Χρέωση δωματίου.		Pass

6.	Test Case Admin forms (Offers management)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
6.1	Επιλογή δωματίου χωρίς να εισάγουμε κάποιο ποσοστό έκπτωσης και πάτημα κουμπιού Save discount	Εμφάνιση μηνύματος σφάλματος για τη μη εισαγωγή έγκυρης τιμής ποσοστού έκπτωσης.		Pass
6.2	Επιλογή δωματίου και εισαγωγή στο πεδίο ποσοστού έκπτωσης αντί αριθμών συμβόλων	Εμφάνιση μηνύματος σφάλματος για τη μη εισαγωγή έγκυρης τιμής ποσοστού έκπτωσης.		Pass
6.3	Επιλογή δωματίου και εισαγωγή στο πεδίο ποσοστού έκπτωσης αντί αριθμών γραμμάτων	Εμφάνιση μηνύματος σφάλματος για τη μη εισαγωγή έγκυρης τιμής ποσοστού έκπτωσης.		Pass

6.4	Επιλογή δωματίου και εισαγωγή στο πεδίο ποσοστού έκπτωσης έγκυρη τιμή.	Εμφάνιση μηνύματος επιτυχούς εισαγωγής και ανανέωση της τιμής του δωματίου βάση τις έκπτωσης.		Pass
------------	--	---	--	------

6.	Test Case Admin forms (Statistics)			
Test #	Tasks	Expected Result	Notes	Pass /Fail
7.1	Πάτημα κουμπιού Export Statistics	Δημιουργία αρχείου xlsx το οποίο θα περιλαμβάνει προσχεδιασμένα στατιστικά.	Δεν υπάρχει φόρμα statistics, ούτε κουμπί Export Statistics. Feature to be implemented	Fail