PDA REPORT

1. Εισαγωγή & Motivation

➢ Τι αφορά το project;

Το παρόν project αφορά την ανάπτυξη ενός PDA Εστίασης, δηλαδή ενός λογισμικού προσωπικού ψηφιακού βοηθού που προορίζεται για χρήση σε εστιατόρια, με σκοπό τη διαχείριση παραγγελιών. Η εφαρμογή είναι υλοποιημένη σε C++ και χρησιμοποιεί τη βιβλιοθήκη Qt για τη δημιουργία γραφικού περιβάλλοντος (GUI), παρέχοντας ένα φιλικό και διαδραστικό περιβάλλον εργασίας για τον χρήστη. Μέσω της εφαρμογής, οι σερβιτόροι μπορούν να καταχωρούν, να διαχειρίζονται και να αποστέλλουν παραγγελίες στην κουζίνα εύκολα και γρήγορα.

➢ Ποιο πρόβλημα λύνει;

Η χρήση χειρόγραφων σημειώσεων ή λεκτικής επικοινωνίας μεταξύ προσωπικού συχνά οδηγεί σε καθυστερήσεις, παρανοήσεις και λάθη στις παραγγελίες. Το συγκεκριμένο σύστημα έρχεται να δώσει λύση σε αυτά τα προβλήματα, προσφέροντας μια ψηφιακή, άμεση και αξιόπιστη εναλλακτική. Η ταχύτητα καταχώρισης, η δυνατότητα επεξεργασίας των παραγγελιών, καθώς και η σαφής οπτική αναπαράστασή τους, ενισχύουν την αποδοτικότητα και μειώνουν τα περιθώρια σφάλματος.

➢ Γιατί επιλέξατε αυτό το αντικείμενο;

Η επιλογή του συγκεκριμένου θέματος έγινε λόγω της πρακτικής του σημασίας και της σύνδεσής του με πραγματικές ανάγκες της αγοράς. Ο τομέας της εστίασης επενδύει όλο και περισσότερο σε τεχνολογικές λύσεις που αυτοματοποιούν και βελτιώνουν την εξυπηρέτηση. Επιπλέον, η ανάπτυξη ενός τέτοιου project σε C++ με Qt αποτελεί μια εξαιρετική ευκαιρία για την εξάσκηση τόσο σε σύγχρονες τεχνικές προγραμματισμού όσο και στη δημιουργία σύγχρονων γραφικών διεπαφών χρήστη.

2. Objective & Scope

➢ Ποιοι είναι οι βασικοί στόχοι;

Βασικός στόχος του project είναι να φτιάξουμε μια εφαρμογή που θα βοηθάει τους σερβιτόρους να καταχωρούν παραγγελίες πιο εύκολα και πιο γρήγορα. Θέλουμε να κάνουμε τη διαδικασία πιο οργανωμένη και να μειώσουμε τα λάθη που γίνονται όταν γράφουν τις παραγγελίες στο χέρι ή τις λένε προφορικά.

Μέσα από αυτό το project, σκοπός μας είναι επίσης να μάθουμε πώς φτιάχνουμε μια εφαρμογή με γραφικό περιβάλλον σε C++, χρησιμοποιώντας τη βιβλιοθήκη Qt. Έτσι, αποκτάμε εμπειρία και σε πιο πρακτικά κομμάτια προγραμματισμού που έχουν εφαρμογή στην καθημερινή ζωή.

➢ Ποιες δυνατότητες περιλαμβάνονται;

- **Διαλέγει τραπέζι και να ξεκινάει μια νέα παραγγελία.**
- **Προσθέτει προϊόντα από το μενού (Brunch,Coffee,Food,Drinks).**
- **Βλέπει τι έχει προσθέσει στην παραγγελία πριν την στείλει.**
- **Εκτυπώνει την απόδειξη όταν ζητηθεί για κάθε τραπέζι ξεχωριστά.**
- **Δέχεται σχόλια σε κάθε προϊόν ξεχωριστά.**
- **Έχει δυνατότητα επεξεργασίας διαφορετικών τραπεζιών ταυτόχρονα.**
- **Αποθηκεύει όλες τις αποδείξεις σε ένα .txt αρχείο.**
- **Έχει εύχρηστο γραφικό περιβάλλον (GUI), ώστε να είναι απλό στη χρήση.**

3. System Architecture

➢ Πώς λειτουργεί το σύστημα;

Η εφαρμογή ξεκινάει με μια οθόνη που δείχνει όλα τα διαθέσιμα τραπέζια (Table 1–9, S1–S9). Ο χρήστης (σερβιτόρος) επιλέγει πρώτα το τραπέζι για το οποίο θέλει να κάνει παραγγελία. Στη συνέχεια, μπορεί να επιλέξει κατηγορία προϊόντων (Brunch, Coffee, Food, Drinks) και να δει τα διαθέσιμα είδη από κάτω σε μια λίστα. Αφού διαλέξει προϊόν και ποσότητα (από το dropdown), πατάει το κουμπί "ADD TO ORDER" για να προστεθεί η επιλογή στη δεξιά λίστα (που φαίνεται σαν "καλάθι" παραγγελίας). Όταν ολοκληρωθεί η παραγγελία, μπορεί να πατήσει "PRINT RECEIPT" για να εκτυπωθεί ή να σταλεί στην κουζίνα.

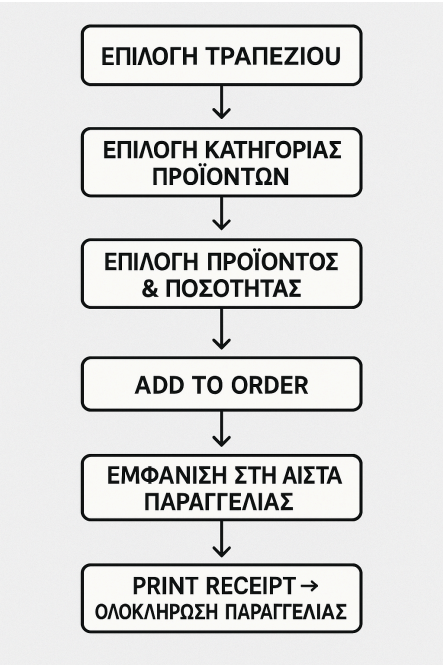
➢ Ποιες είναι οι βασικές του συνιστώσες;

<u>Βασικά μέρη της εφαρμογής είναι:</u>

- **Κουμπιά Τραπεζιών** επιλέγεις το τραπέζι για το οποίο γίνεται η παραγγελία.
- **Κατηγορίες Μενού** (Brunch, Coffee, Food, Drinks) φιλτράρουν τα προϊόντα.
- **Λίστα Προϊόντων** δείχνει τα διαθέσιμα είδη για επιλογή.
- **Dropdown Ποσότητας** επιλέγεις πόσα τεμάχια θα προστεθούν.

- **Κουμπί "ADD TO ORDER"** προσθέτει προϊόντα στη δεξιά λίστα.
- **Περιοχή Προεπισκόπησης Παραγγελίας** εμφανίζει τι έχει προστεθεί.
- **Κουμπί "PRINT RECEIPT"** την εκτυπώνει.

➢ **Διάγραμμα ροής**



4. Τεχνολογίες που Χρησιμοποιήθηκαν

➢ **Ποιες και γιατί;**

Για την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε C++ σε συνδυασμό με το Qt Framework, γιατί μας επιτρέπει να δημιουργήσουμε εύκολα γραφικά περιβάλλοντα (GUI) και να χειριστούμε παράθυρα, κουμπιά και άλλα στοιχεία της εφαρμογής. Επίσης, είναι κατάλληλο για cross-platform εφαρμογές και έχει πολλές έτοιμες δυνατότητες για desktop προγράμματα.

Γιατί Qt:

- **Είναι φιλικό προς τον προγραμματιστή, ειδικά για GUI.**
- **Υποστηρίζει drag & drop σχεδίαση παραθύρων με το Qt Designer.**
- **Έχει πολλές έτοιμες βιβλιοθήκες για αρχεία, δικτύωση, χειρισμό συμβάντων κ.ά.**

- **Συνδυάζεται άνετα με την C++, που είναι η κύρια γλώσσα του project.**

➢ **Αναφορά βιβλιοθηκών που χρησιμοποιηθηκαν:**

- **libgcc_s_seh-1.dll:** Βιβλιοθήκη runtime της GCC που υποστηρίζει τη διαχείριση εξαιρέσεων (structured exception handling) στα Windows.
- **libstdc++-6.dll:** Η βασική βιβλιοθήκη για τη C++ standard library.
- **libwinpthread-1.dll :**Υποστηρίζει πολυνηματική εκτέλεση (multithreading) σε Windows.
- **opengl32sw.dll:** Βιβλιοθήκη για software rendering OpenGL, που χρησιμοποιείται ως εναλλακτική όταν το hardware acceleration δεν είναι διαθέσιμο.
- **Qt6Core.dll:** Η βασική βιβλιοθήκη του Qt που χειρίζεται τα θεμελιώδη στοιχεία (όπως χρονισμός, σήματα και υποδοχείς).
- **Qt6Gui.dll:** Αφορά τα γραφικά στοιχεία της εφαρμογής (εικόνες, παραθυρικά στοιχεία).
- **Qt6Network.dll:** Υποστηρίζει λειτουργίες δικτύωσης, σε περίπτωση που η εφαρμογή γίνει δικτυακή στο μέλλον.
- **Qt6Svg.dll:** Χρησιμοποιείται για εμφάνιση εικόνων SVG (αν και δεν χρησιμοποιήθηκε άμεσα).
- **Qt6Widgets.dll:** Περιλαμβάνει τα βασικά GUI widgets, όπως κουμπιά, λίστες, πεδία κειμένου κ.λπ.

5. Κώδικας & Υλοποίηση

➢ **Πώς υλοποιήθηκε η βασική λειτουργικότητα;**

Η βασική λειτουργικότητα του συστήματος παραγγελιοληψίας για καφετέρια έχει υλοποιηθεί με τη χρήση της βιβλιοθήκης Qt (σε C++), αξιοποιώντας τις δυνατότητες του GUI framework για την ανάπτυξη γραφικής διεπαφής και λογικής εφαρμογής.

<u>Συγκεκριμένα, η εφαρμογή:</u>

- **Φορτώνει το μενού προϊόντων στην loadMenu() με κατηγοριοποίηση (Brunch, Coffee, Food, Drinks).**

- **Επιτρέπει την επιλογή τραπεζιού και δημιουργία/φόρτωση αντίστοιχης παραγγελίας.**
- **Επιτρέπει την προβολή προϊόντων ανά κατηγορία, προσθήκη προϊόντων στην παραγγελία με σχόλια και ποσότητα.**
- **Υποστηρίζει εκτύπωση απόδειξης ανά τραπέζι ξεχωριστά και με καταγραφή στο αρχείο orders.txt από όλες τις παραγγελίες μαζί που έγιναν**

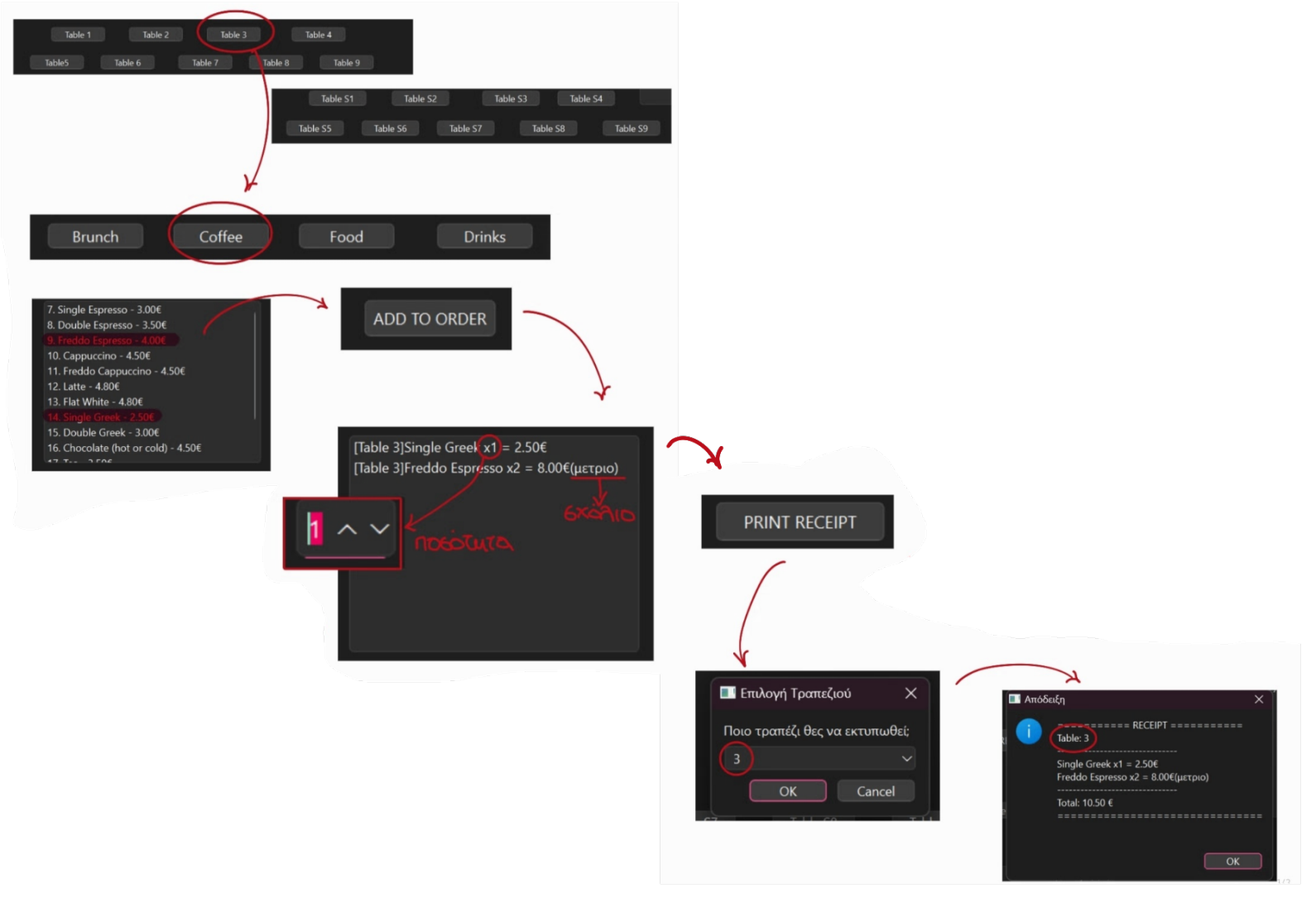
➢ **Υπάρχουν σημαντικοί αλγόριθμοι ή δομές δεδομένων;**

Σημαντικές δομές δεδομένων:

- **QList<Product>:** Μενού προϊόντων
- **QMap<QString, Order>:** Παραγγελίες ανά τραπέζι
- **QString:** Επιλεγμένο τραπέζι, σχόλια κ.ά.
- **QListWidgetItem:** Αντικείμενα λίστας για εμφάνιση UI
- **QStringList:** Λίστα IDs τραπεζιών
- **QFile & QTextStream:** Εγγραφή απόδειξης σε αρχείο

➢ **REPO:**https://github.com/christsi33/PdaGui

6. Αποτελέσματα & Demo



7. Comparison with AI generated code

Για τη σύγκριση, επέλεξα δύο βασικές λειτουργίες της εφαρμογής: την προσθήκη παραγγελίας και την εκτύπωση απόδειξης. Ο κώδικας του AI ήταν λειτουργικός, αλλά πιο απλοποιημένος σε σχέση με τον δικό μου.

AI:

```
connect(ui->btnAddToOrder, &QPushButton::clicked, this, [this]() {
    QListWidgetItem* item = ui->listWidget->currentItem();
    if (!item) return;

    int id = item->data(Qt::UserRole).toInt();
    int qty = ui->spinQuantity->value();
    if (qty <= 0) return;

    QString comment = ui->lineComm->text();

    // Απλά παίρνουμε το πρώτο προϊόν με το id (χωρίς for loop)
    Product prod;
    bool found = false;
    for (auto &p : menu) {
        if (p.id == id) {
            prod = p;
            found = true;
            break;
        }
    }
    if (!found) return;

    OrderItem orderItem(prod, qty, comment);
    orders[selectedTable].append(orderItem);

    ui->listWidgetOrder->addItem(QString("Table %1:
%2").arg(selectedTable).arg(orderItem.toString()));

    ui->lineComm->clear();
});

connect(ui->btnPrintReceipt, &QPushButton::clicked, this, [this]() {
    if (orders.isEmpty()) {
        QMessageBox::information(this, "No Orders", "No active
orders.");
        return;
    }

    // Εμφάνιση απλής λίστας τραπεζιών με παραγγελίες χωρίς φίλτρο
    QStringList tables;
    for (const QString& t : orders.keys()) {
        tables << t;
    }
```

```
    if (tables.isEmpty()) {
        QMessageBox::information(this, "No Orders", "No orders to
print.");
        return;
    }

    QString table = tables.first(); // Επιλογή απλά του πρώτου
τραπεζιού

    QString receipt = "Receipt for Table " + table + "\n";
    for (const OrderItem& oi : orders[table]) {
        receipt += oi.toString() + "\n";
    }

    QMessageBox::information(this, "Receipt", receipt);
});
```

MY CODE:

```
connect(ui->btnAddToOrder, &QPushButton::clicked,this,[this](){
        QListWidgetItem* selectedItem = ui->listWidget->currentItem();
        if(!selectedItem) return;

        int id = selectedItem->data(Qt::UserRole).toInt();

        for(const Product& p: menu){
            if(p.id == id){
                int quantity = ui->spinQuantity->value();
                if(quantity<=0){
                    QMessageBox::warning(this,"Attention","Quantity
must be grater then 0!");
                    return;
                }
                QString comment = ui->lineComm->text();
                OrderItem item(p,quantity,comment);
                orders[selectedTable].addItem(item);

                ui->listWidgetOrder->addItem(
                    QString("[Table %1]%2").arg(selectedTable,
item.toString()));
                break;
            }
        }
        ui->lineComm->clear();
    });

    connect(ui->btnPrintReceipt, &QPushButton::clicked, this, [this]()
{
        if (orders.isEmpty()) {
            QMessageBox::information(this, "No Orders", "There are no
active orders.");
            return;
```

```
        }

        // Δημιουργούμε λίστα με τα ανοιχτά τραπέζια
        QStringList tableIds;
        for (const QString& key : orders.keys()) {
            if (!orders[key].isEmpty()) {
                tableIds.append(key);
            }
        }

        if (tableIds.isEmpty()) {
            QMessageBox::information(this, "No Active Orders", "There
are no open orders to print.");
            return;
        }

        // Ζητάμε από τον χρήστη να επιλέξει τραπέζι
        bool ok;
```

1. Προσθήκη παραγγελίας:
Και οι δύο υλοποιήσεις αναζητούν το προϊόν με βάση το id μέσω for loop και προσθέτουν την παραγγελία. Ωστόσο, ο δικός μου κώδικας ελέγχει αν η ποσότητα είναι έγκυρη και εμφανίζει προειδοποιητικό μήνυμα, ενώ ο AI κώδικας απλώς επιστρέφει χωρίς καμία ειδοποίηση. Αυτό κάνει τη δική μου υλοποίηση πιο φιλική προς τον χρήστη.

2. Εκτύπωση απόδειξης:
Ο AI κώδικας εκτυπώνει απόδειξη για το πρώτο διαθέσιμο τραπέζι χωρίς επιλογή. Αντίθετα, ο δικός μου φιλτράρει μόνο τα τραπέζια με ενεργές παραγγελίες και προβλέπει την επιλογή τραπεζιού από τον χρήστη (με διαδραστικό τρόπο). Έτσι προσφέρει μεγαλύτερη ευελιξία και ακρίβεια.

Συμπερασματικά:
Ο AI-generated κώδικας είναι πιο συνοπτικός και γρήγορος στην υλοποίηση, αλλά με λιγότερους ελέγχους. Ο δικός μου κώδικας είναι πιο πλήρης, πιο ασφαλής και προσφέρει καλύτερη εμπειρία χρήστη, ιδιαίτερα σε σενάρια με σφάλματα ή πολλαπλές παραγγελίες.

8. Conclusions & Lessons learned

Συμπεράσματα

Μέσα από αυτή την εργασία καταφέραμε να φτιάξουμε μια απλή και λειτουργική εφαρμογή που μπορεί να χρησιμοποιηθεί σε μαγαζί εστίασης για την καταγραφή παραγγελιών. Παρόλο που στην αρχή δεν ήμασταν πολύ εξοικειωμένοι με το Qt, τελικά καταφέραμε να το κατανοήσουμε και να το χρησιμοποιήσουμε για να φτιάξουμε ένα καθαρό περιβάλλον χρήστη. Αντιμετωπίσαμε κάποια μικρά προβλήματα στην πορεία, αλλά με δοκιμές και ψάξιμο βρήκαμε λύσεις και φτάσαμε σε ένα καλό αποτέλεσμα.

Τι μάθαμε

- Πώς να χρησιμοποιούμε Qt για GUI εφαρμογές σε C++.
- Πώς να σχεδιάζουμε δομές δεδομένων για παραγγελίες, μενού, τραπέζια κ.λπ.
- Πώς να δουλεύουμε με πολλαπλά components και να τα "δένουμε" μεταξύ τους.
- Πόσο σημαντική είναι η καλή οργάνωση κώδικα και η διαχείριση γεγονότων (signals/slots).
- Την αξία της δοκιμής σε πραγματικό περιβάλλον και του να βλέπεις τι δουλεύει και τι όχι στην πράξη.