***Documentation of: TikiPark Mobile Parking App***

for: 1) Έγγραφο Απαιτήσεων, 2) Εγχειρίδιο Χρήσης

# Detailed documentation of System:

## Απαιτήσεις

**Λειτουργικές Απαιτήσεις**

* Δημιουργία/Σύνδεση Χρήστη
* Λειτουργίες Admin
* Κράτηση θέσης στάθμευσης
* Υπολογισμός κόστους βάσει χρόνου
* Ολοκλήρωση πληρωμής
* Στατιστικά χρήστη και ολικά

**Μη Λειτουργικές Απαιτήσεις**

* Απόκριση εφαρμογής κάτω από 2 δευτερόλεπτα
* Αντοχή σε ταυτόχρονες κρατήσεις
* Εύχρηστο περιβάλλον χρήστη (UI/UX)
* Υψηλό επίπεδο ασφαλείας για προσωπικά δεδομένα

## Τεχνολογίες και Αρχιτεκτονική Εφαρμογής

**Tech Stack**

* Τεχνολογίες που χρησιμοποιούνται: Android Studio (Java + XML): Frontend ανάπτυξη εφαρμογής Android.
* Java & PHP: Backend λογική και διαχείριση αιτημάτων.
* MySQL: Βάση δεδομένων για αποθήκευση δεδομένων χρήστη, κρατήσεων κτλ.
* XAMPP (Apache Server): Τοπικός server για την φιλοξενία του PHP backend.
* Google Maps API: Ενσωμάτωση χαρτών και γεωγραφικών λειτουργιών στην εφαρμογή.

**Επικοινωνία με το Backend**

* Η επικοινωνία μεταξύ της εφαρμογής Android και του backend γίνεται μέσω HTTP αιτημάτων. Για την υλοποίηση της δικτυακής κλήσης χρησιμοποιείται:
* Thread + HttpURLConnection

**Γιατί χρησιμοποιούμε Thread + HttpURLConnection:**

* Απλότητα και Μη Εξάρτηση από Εξωτερικά Libraries
* Χρησιμοποιούμε κώδικα native της Java, χωρίς να εισάγουμε επιπλέον εξωτερικά dependencies (όπως το OkHttp ή Retrofit).
* Αυτό διατηρεί το project ελαφρύ και απλό, χωρίς ανάγκη για πρόσθετες βιβλιοθήκες.
* Ασύγχρονη Εκτέλεση για Αποφυγή Μπλοκαρίσματος UI Thread
* Οι δικτυακές κλήσεις εκτελούνται σε ξεχωριστό thread ώστε να μην παγώνει η διεπαφή χρήστη.
* Αυτό είναι υποχρεωτικό στο Android για ομαλή εμπειρία χρήστη και αποφυγή ANR (Application Not Responding).
* Πλήρης Έλεγχος στη Διαδικασία της Δικτυακής Κλήσης
* Με το HttpURLConnection έχουμε άμεση πρόσβαση και διαχείριση όλων των σταδίων του αιτήματος (άνοιγμα σύνδεσης, ρυθμίσεις headers, διαχείριση response, χειρισμός errors).
* Παρέχεται η δυνατότητα λεπτομερούς παραμετροποίησης και debugging σε κάθε βήμα.

**Σύντομη Αναφορά Υλοποίησης**

* Δημιουργείται νέο Thread ή χρησιμοποιείται AsyncTask (σε παλαιότερες υλοποιήσεις) για την εκτέλεση της δικτυακής κλήσης.
* Μέσα στο thread, με χρήση του HttpURLConnection ανοίγεται σύνδεση προς το backend URL.
* Στέλνονται δεδομένα (π.χ. POST parameters) και λαμβάνεται η απόκριση.
* Η απόκριση διαβάζεται και αποστέλλεται πίσω στο UI thread για ενημέρωση της διεπαφής χρήστη.

## Αρχιτεκτονική Συστήματος:

**Σχεδιαστικές Αποφάσεις**

* Χρήση Android UI components και Material Design για μοντέρνα και συνεπή εμπειρία χρήστη.
* Caching τοπικών δεδομένων (π.χ. μέσω SharedPreferences) για άμεση φόρτωση χωρίς εξάρτηση από το backend.
* Ασύγχρονη επικοινωνία με τη MySQL μέσω PHP queries και prepare() statements για αποφυγή μπλοκαρίσματος.
* Intent-based navigation μεταξύ Activities για σαφή ροή δεδομένων και διαχωρισμό λειτουργιών.
* Ενσωμάτωση JSON ως φορμά επικοινωνίας μεταξύ Android και PHP για συμβατότητα και ευκολία ανάλυσης.
* Διαχείριση αποκρίσεων μέσω parsing και χρήσης Java μοντέλων (POJOs) για καθαρό διαχωρισμό UI από τα δεδομένα.
* Modular σχεδίαση backend μέσω require\_once για κοινά modules (π.χ. config.php) και αποφυγή επανάληψης.
* Custom Threads με χρήση HttpURLConnection για HTTP αιτήματα χωρίς εξαρτήσεις από τρίτες βιβλιοθήκες.
* Ενημέρωση UI μέσω Handlers που λαμβάνουν αποτελέσματα από τα δευτερεύοντα threads.
* Δομημένος χειρισμός σφαλμάτων μέσω try-catch και ασφαλών JSON αποκρίσεων (π.χ. json\_encode(["error" => "..."])).
* Logging τόσο στην Android πλευρά (Logcat) όσο και στην PHP πλευρά (echo/debug output) για ανάπτυξη και debugging.
* Κατάσταση των θέσεων parking διαχειρίζεται μέσω flags στη βάση δεδομένων (available, reserved, occupied) για αποφυγή σύγκρουσης κρατήσεων.

**Πρακτικές Ασφάλειας**

* Χρήση prepare() σε όλα τα SQL queries για πρόληψη SQL Injection.
* Αποθήκευση κωδικών με χρήση της password\_hash() και επαλήθευση με password\_verify() για ασφαλή διαχείριση πιστοποίησης.
* Έλεγχος εισερχόμενων παραμέτρων μέσω isset() και του τελεστή ?? για αποφυγή απροσδόκητων σφαλμάτων και injection attempts.
* Μηνύματα λάθους σε μορφή JSON με γενικές περιγραφές για αποτροπή διαρροής πληροφοριών του server ή της βάσης.
* Διαχείριση της κατάστασης πληρωμών μέσω flags (payment\_status: pending, paid) για σταθερή παρακολούθηση συναλλαγών.
* Δυνατότητα επέκτασης με session ελέγχους ή χρήση API keys για περιορισμό πρόσβασης σε κρίσιμα scripts.
* Ορισμός HTTP headers (όπως Content-Type: application/json) και χρήση CORS policies για ασφαλή επικοινωνία μεταξύ Android εφαρμογής και PHP backend.

**Java Best Practices**

* Διαχωρισμός Κώδικα: Χρήση MVC (Model-View-Controller) νοοτροπίας, με model classes (POJOs), network handlers και UI components χωριστά.
* Avoid Magic Strings: Χρήση σταθερών (public static final) για URLs, keys, flags αντί για hardcoded strings.
* Null Safety: Προληπτικός έλεγχος για null τιμές (if (obj != null) {...}) για αποφυγή runtime crashes.
* Exception Handling: Χρήση try-catch μπλοκ για δικτυακές και μετατροπες λειτουργίες (e.g. IOException, JSONException).
* Reusability: Δημιουργία helper μεθόδων για κοινές λειτουργίες (π.χ. HTTP requests, response parsing).
* Thread Management: Αποφυγή βαριών διεργασιών στο UI thread – χρήση Thread, Handler, Runnable.
* Logging: Καταγραφή debug/σφαλμάτων με Log.d(), Log.e() για εντοπισμό στη διάρκεια ανάπτυξης.
* Modularity: Σπασμένα components σε κλάσεις και φακέλους (π.χ. models, network, utils) για ευκολία πλοήγησης.
* Καθαρός Κώδικας (Clean Code): Περιγραφικά ονόματα μεταβλητών/μεθόδων, αποφυγή long methods, σωστό indentation.

**Πρακτικές Ομαδικής Ανάπτυξης (Team Development Practices)**

* Version Control: Χρήση GitHub για καταγραφή αλλαγών, versioning, rollback και συνεργασία.
* Branching Strategy: Δημιουργία branches για κάθε feature/bug fix με merge μόνο μετά από έλεγχο.
* Κατανομή Ρόλων: Ξεκάθαρος διαχωρισμός ευθυνών ανά μέλος για Front-end, Backend, UI, Testing.
* Code Reviews: Έλεγχος κώδικα μεταξύ μελών για ποιότητα, bug detection και γνώση της βάσης κώδικα.
* Testing: Επαλήθευση της λειτουργικότητας σε τοπικό περιβάλλον πριν γίνει merge/παράδοση.
* Documentation: Συνοδευτική τεκμηρίωση κάθε module (frontend/backend) για ευκολία συντήρησης.
* Συναντήσεις Ομάδας: Τακτικός συντονισμός προόδου (διά ζώσης ή online), με καθορισμό milestone στόχων.

**Μέλη Ομάδας:**

*Κωφίδης Θεόφιλος (iis23028): Analysis*

*Σταυρή Ουρανία-Ραφαηλία (iis23066): MockUps, UI*

*Μπαλόγλου Πέτρος (ics23024): Design, Deliverables*

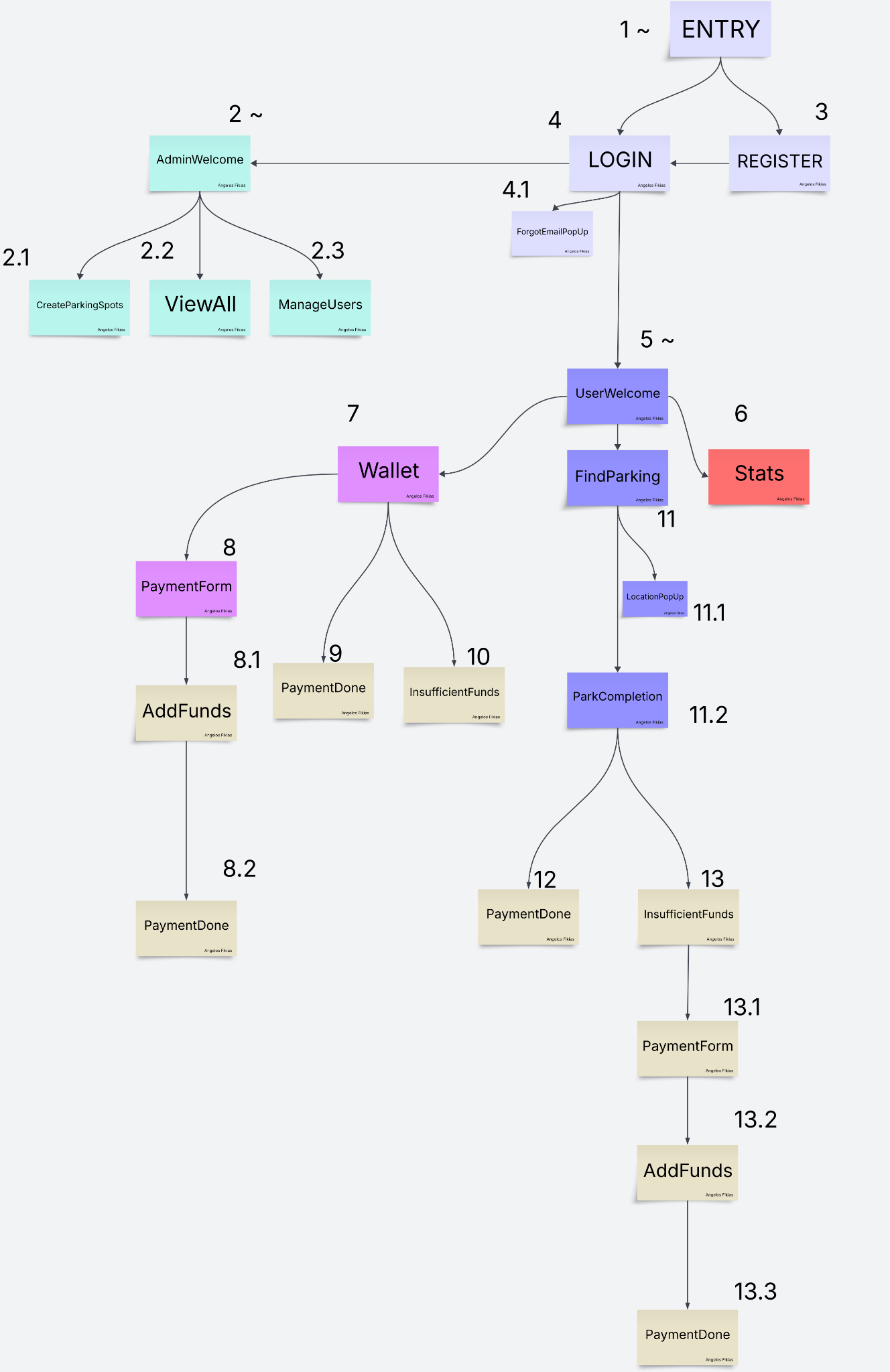
*Στεφανίδης Νικόλαος (ics23003) : Front-end, UI*

*Πετσιμέρης Φοίβος-Εμμανουήλ (ics23125): APIs, Classes, Testing*

*Φίκιας Άγγελος (iis23006): Back-end*

# Step By Step Tutorial

for: 1) YoutubeVid, 2) Presentation



# ScreenShots

-> …

# vid

-> …

# Παραδοτέα:

1. Εγγραφο Απαιτήσεων (παρουσίαση ανταγωνισμού, λειτουργικές απαιτήσεις, mockups) + Κώδικας της Εφαρμογής (GitHub link)
2. Video παρουσίαση της εφαρμογής
3. Εγχειρίδιο χρήσης με screenshot + μέσω YouTube link