ChatGPT

تحليل خطة Manus التطويرية

نظرة عامة على الخطة

تهدف خطة التطوير إلى رفع جودة تطبيق *Delivery Ways* لمستوى مشابِه لمنصات كـUber من خلال تنفيذ أربعة مسارات رئيسية (Tracks A–D) تركز على مختلف جوانب التطبيق، تليها مرحلة تنفيذ نهائية. تتضمّن المسارات: **A** (نظام التصميم وقالب التطبيق)، **B** (تطبيق ميزات الرحلات)، **C** (التطبيق الأساسي للطُرُد والطعام)، **D** (التفعيل: الإعداد/التوثيق/DSR). ترتكز الخطة على شروط هندسية صارمة مثل تمرير جميع الاختبارات الـ158 الحالية، والمحافظة على هيكلية الشيمز الحالية، وعدم استيراد أي SDK خارجي مباشرة في `app/lib` [1].

Track	العنوان في الخطة	الهدف	أهم الـ Deliverables
A	*Design System & App Shell Implementation*	توحيد نظام التصميم (Theme + Components) وإنشاء قالب (AppShell) موحّد للتطبيق.	- مكون `AppShell` موحّد (scaffold عام) - تطبيق ثيم `Material3` موحّد عبر التطبيق - مكتبة مكونات واجهة موحّدة (أزرار، بطاقات، شريط تبويب سفلي، إلخ)
B	*Ride Vertical MVP Implementation*	بناء تجربة حجز رحلة كاملة (المكوّنات الأمامية) بشكل MVP.	- شاشات اختيار المواقع (التقاط/انزال) مع خريطة - شاشة حجز الرحلة مع تفاصيل الأجرة والزمن المقدر - شاشة تتبع الرحلة في الوقت الحقيقي (خريطة، حالة السائق) - شاشة إنهاء الرحلة مع ملخص، تقييم السائق، وإيصال الرحلة
C	*Parcels & Food MVP Implementation*	تفعيل خدمات الطرود والطعام كـMVP مع دعم فائق للأولى وخيارية للأكل.	- إدارة الطرود: حالات, إنشاء وتتبع الطرود - شاشات الطرود (قائمة الطرود، تفاصيل الطرد، إنشاء طرد)

Track	العنوان في الخطة	الهدف	أهم الـ Deliverables
			- إضافة Flag للخدمة الغذائية في الـFeatureFlags ودعم إخراج "قريباً" إن تعطلت
D	*Onboarding, Auth and DSR Implementation*	استكمال تدفق الإعداد للمستخدم (Onboarding)، تفعيل شاشات التسجيل/التحقق، وإنشاء شاشات حقوق البيانات (DSR).	- شاشات Onboarding جديدة (ترحيب، أذونات، تفضيلات)
			- تفعيل شاشات المصادقة (الدخول برقم الهاتف، رمز التحقق، 2FA)
			- شاشات حالات فارغة وأخطاء (بدون رحلات/طرود)
			- شاشات DSR (طلب تصدير/حذف البيانات وفق اللوائح)

القيود الصارمة:
- **الاختبارات:** يجب أن تمرّ جميع الاختبارات الـ158 الحالية أو تُحدَّث للحفاظ على مستوى التغطية الحالي (مطابقة لما لدينا من اختبارات موجودة).
- **البنية:** المحافظة على هيكلية الشيمز/Packages الحالية (`mobility_shims`, `payments`, `foundation_shims`, إلخ) كما هي.
- **عدم استيراد الـSDK مباشرة:** منع استيراد أي حزمة خارجية (Stripe, Google Maps, Firebase إلخ) داخل `app/` و`lib` — يجب أن تتم كل التكاملات من خلال طبقات الشيمز [1].
- **سلامة البناء:** يجب أن يظل المشروع قابلًا للبناء والتشغيل باستخدام الأوامر المعيارية (`flutter pub get`, `flutter test`, `flutter build apk --debug`, `flutter build ios --no-codesign`).

حالة المشروع v3.2.1 مقارنةً بالوضع الحالي: يغطي تقرير الحالة (v3.2.1) جانبًا محددًا وهو ميزة الإشعارات في طبقة الواجهة (Notifications feature) [2]. لا يتناول التقرير مسارات الخطة (A–D) بشكل مباشر، ولا يبدو أن هناك تعارضًا بين ما ينفذه التقرير (إشعارات العميل) ووضع `DW_workspace` الحالي. بحالته الراهنة، تعتبر ميزة الإشعارات "جاهزة للعميل" (الواجهة جاهزة مع انتظار الربط بالباكند) [2]. الخطة لا تعتمد على عمل سابق لميزة الإشعارات، لكنها يجب أن تراعي مكتسبات التصميم المشترك (مثل استخدام شاشات التحميل والفارغة) المطبقة هناك. باختصار، نطاق التقرير هو الإشعارات فقط، والتطبيق الحالي (وهيكله وبنيته) يحترم نفس معايير Clean-B (غياب استيراد SDK وبنية الشيمز) [1].

تحليل الفجوات لكل Track

Track A: نظام التصميم وقالب التطبيق

TL;DR: يهدف هذا المسار إلى إرساء نظام تصميم موحد (Colors, Typography, Spacing) وإنشاء قالب التطبيق العام (`AppShell`) ليضمّن شريط تنقل سفلي/عُلوي موحدًا. الخطة تفترض وجود أساس للتصميم (مثل `AppThemeData` وألوان/أنماط) وتحتاج لتنشيطه وتوحيده في جميع الشاشات.

- **ما يفترضه Manus عن الوضع الحالي:**
- لدينا أساس Theme قائم (`AppThemeData` مع ألوان أساسية)، وأنماط نص واضحة، ونظام تباعد Grid بقيم محددة، ومكونات مسبقة في B-UI (الأزرار، البطاقات، حالات التحميل).
- لا يوجد *AppShell* موحّد حاليًا؛ التطبيق يعتمد على `ColorScheme.fromSeed` في مواضع متعددة.
- الشاشات الحالية تستخدم طرق متفرقة للتصميم (بعضها يستورد مواد Material مباشرة).

- بنية الشيمز الخاصة بنظام التصميم موجودة، لكن غير مفعّلة في الكود حاليًا.

- **الواقع الحالي (DW_workspace):**

- **جزئيًا مُنفّذ:** بنية نظام التصميم موجودة جزئيًا (حزم `design_system_shims/components/` `foundation` متصلة بالتطبيق) [3] ، لكن غالبية الشاشات لم تنتقل لاستخدامها بعد.
- التطبيق يستخدم أحيانًا `ColorScheme.fromSeed` وثيمات Material3 مباشرة بدلًا من `AppThemeData` .
- لا يوجد ملف أو كود حالي لـ `AppShell` موحّد. الشاشات تستخدم Scaffold مباشرةً معbar غير موحّد.

- بعض المكونات الموحدة (B-UI) متوفرة، لكن الشاشات الحالية لا تستفيد منها بالكامل.

- **الفجوات والتعارضات:**

- الفجوة الأهم: **عدم وجود AppShell** موحّد؛ يجب إنشاؤه من الصفر.
- اعتماد التطبيق الحالي على ثيمات Material مباشرة يتعارض مع الفرضية أنه يستخدم `AppThemeData` .
- تباين في استخدام المكونات وسمات الواجهة قد يؤدي إلى تصاميم غير موحّدة (فجوة في الاتساق التصميمي).

- لا تعارض مع قواعد Clean-B؛ بل على العكس، الخطة تدعمها (استخدام الثيم والستايليست الخاص، لا استيراد SDK) [2] [1] .

- **إنجازات سريعة vs عمل ثقيل:**

- **إنجازات سريعة:**
  - تفعيل استخدام `AppThemeData` في `MaterialApp` (إلغاء `ColorScheme.fromSeed`)، حيث الخلايا اللونية والعناوين جاهزة.
  - تطبيق `theme.spacing` والتصميمات الموحدة في عدد من الشاشات (خطوة بداية بسيطة بنقل الـProvider).
- **عمل ثقيل:**
  - بناء كامل قالب `AppShell` (مكونات التنقل والتجاوب حسب حجم الشاشة).
  - إعادة كتابة شاشات متعددة لاستخدام `AppShell` والمكونات الموحدة (قد يتطلب تعديل هرمي كبير للواجهات).

Track B: تدفق الرحلات (Ride Vertical)

TL;DR: يركّز هذا المسار على إكمال تجربة حجز الرحلة نهايةً إلى نهاية (MVP). يتضمن إنشاء شاشات لاختيار موقع التقاط/إنزال، وشاشة تلخيص الحجز (مع تقدير الأجرة)، وشاشة تتبع الرحلة، وشاشة إنهاء الرحلة مع ملخص للسائق والأجرة.

- **ما يفترضه Manus عن الوضع الحالي:**
- حزم شيمز خاصة بالحركة موجودة (`mobility_shims` لمسار الركوب و`maps_shims` للخرائط) توفر واجهات مجردة لإدارة الرحلات وعرض الخرائط.
- توجد شاشات تتبع أساسية: `tracking_screen.dart` و`tracking_map_screen.dart` و`order_tracking_screen.dart`.
- لا يوجد حالياً تدفق حجز مكتمل؛ على الأرجح تفتقر الشاشات إلى اختيار المواقع، وحساب الأجرة.
- يُفترض وجود نظام معاملات مشتركة (مثل التصاريح، التحويلات) في الخلفية جاهزة للربط.
- **الواقع الحالي (DW_workspace):**
- لدينا حزم `mobility_shims` و`maps_shims` مُفعّلة ومتصلة بالتطبيق [3]، والبنية الأساسية للخرائط موجودة.
- توجد شاشات تتبع للرحلات قيد الاستخدام وبعض اختبارات واجهة (`tracking_screen_test.dart`)، مما يدل على وجود هيكل للتتبع.
- لا توجد حاليًا شاشات اختيار المواقع (لم تُنفذ)، ولا شاشات لحساب الأجرة (fare estimation)، ولا شاشة لإنهاء الرحلة (خارج مسار الطلبات).
- سيحتاج الاندماج مع باكند الرحلات (API) وهذا غير مفعل بالكامل؛ على الأغلب يعتمد على stubs حاليًا.
- **الفجوات والتعارضات:**
- الفجوة الأساسية: **غياب واجهات الـ UI** الخاصة بالحجز (اختيار المواقع وعرض الأسعار).
- الشاشات الموجودة تركز على التتبع فقط، ولا تغطي المسار من البداية.
- لا تعارض مع القواعد الحالية؛ البناء يتم عبر الشيمز الموجودة والسماح للتطوير التدريجي.
- قد يكون هناك تضارب في المكونات الموحدة (يجب استخدام `AppShell` من المسار A مطبق سابقًا).
- **إنجازات سريعة vs عمل ثقيل:**
- **إنجازات سريعة:**
  - إعادة استخدام `maps_shims` لعرض الخريطة في شاشات اختيار المواقع (بإضافة مكان مدخل).
  - استخدام الحزمة الحالية لحساب الأجرة (أو stub) لعرض التقدير الأولي.
- **عمل ثقيل:**
  - بناء شاشة حجز كاملة (`RideBookingScreen`) مع كل الحقول والتفاعلات.
  - ضبط State management وربط الشاشات الجديدة بحزم الـshims والـproviders (خاصة إذا تطلب الأمر إدارة حالات الرحلة الكاملة).

Track C: طرود وطعام (Parcels & Food)

TL;DR: يركّز هذا المسار على تفعيل خدمة الطرود كاملةً وضمان وجود Service Flag للخدمة الغذائية (Food Delivery) كخيّار مستقبلي. المطلوب إنشاء إدارة حالة للطرود، وشاشات قائمة الطرود، وتفاصيلها، وإنشاء طرد جديد، بالإضافة إلى تطبيق Flag فعال أو قائمة انتظار للطعام.

- **ما يفترضه Manus عن الوضع الحالي:**
- حزمة `parcels_shims` موجودة لتوفير واجهات مجردة لإدارة الطرود.
- هناك شاشات موضعية (Placeholders) تظهر خيار الطرود والطعام دون وظائف فعلية.
- نظام الـFeature Flags موجود للتبديل بين تفعيل ميزة Food أو إخفائها.
- Backend لطلبات الطرود والطعام قد يكون غير متوفر أو على شكل مفاهيمي فقط.
- **الواقع الحالي (DW_workspace):**
- **حزمة الطرود مفقودة**: لا يوجد في المستودع الحالي حزمة `parcels_shims` كما تفترض الخطة (فجوة صريحة).
- توجد بعض الشاشات التجريبية أو القائمة لإدارة الطرود/الطعام بحسب وصف "شاشات demo". لكنها غير مكتملة وظيفيًا.
- نظام الـFeature Flags موجود ويرتكز على ملفات إعداد (بما فيها على الأغلب `enableFoodDelivery`).
- في حالة عدم وجود حزمة شيمز للطرود، فهذا يفتح فجوة تحتاج معالجتها.
- **الفجوات والتعارضات:**
- **غياب** `parcels_shims` في الكود الحالي، بالرغم من افتراض الخطة لوجودها. يجب إنشاؤها أو تعديل الخطة لاستخدام بنية أخرى.
- الشاشات النائبة للطرود/الطعام موجودة فقط كموديلات عرض؛ ليس لها وظائف حقيقية (الفجوة في التنفيذ).
- لا تعارض مع قواعدنا؛ المبادئ (شيمز فقط، البنكيّات) يجب الحفاظ عليها.
- لا يوجد تعارض في التصميم أو التجريب، فقط نقص في التنفيذ والتهيئة.
- **إنجازات سريعة vs عمل ثقيل:**
- **إنجازات سريعة:**
  - تفعيل نظام Feature Flag لإظهار/إخفاء خيار الطعام على الشاشة الرئيسية اعتمادًا على Flag الموجود.
  - إعادة استخدام حزم شيمز موجودة (مثل `parcels_shims` إذا تمت إضافتها بسرعة) لربط البيانات الوهمية بالشاشات.
- **عمل ثقيل:**
  - تطوير كامل تدفق إدارة الطرود (State Management، شاشات إدخال المواقع/الحجم/الوزن).
  - كتابة شاشات UI متعددة (قائمة الطرود، التفاصيل، الإنشاء) وربطها بالـproviders.

Track D: الإعداد والتوثيق وDSR

TL;DR: يغطي هذا المسار استكمال تدفق إعداد المستخدم (Onboarding screens)، تفعيل شاشات التوثيق (Phone login/OTP/2FA)، وإنشاء شاشات طلبات حقوق البيانات (DSR: التصدير/الحذف).

- **ما يفترضه Manus عن الوضع الحالي:**

- توجد شاشات مصادقة أساسية (OTP، 2FA، تسجيل برقم الهاتف) تم إنشاؤها أو وضعها كمسودات.
- توجد شاشة Onboarding جذرية بشكل مبدئي فقط.
- نظام Feature Flags يدير تفعيل بعض الخطوات (مثلاً تظهر شاشة التوجيه فقط إذا وجدت Flag).

- دعم اللغات (EN/DE/AR) معدّ في التطبيق (ملفات ARB مكتملة) [2] .

- **الواقع الحالي (DW_workspace):**

- **جزئيًا منفّذ:** الشاشات اللازمة للمصادقة موجودة ضمن مسارات Settings/Auth (بما في ذلك شاشتي التحقق وإدخال الهاتف)، لكنها قد تحتاج ربطًا ببيانات فعلية. تقرير الحالة يوضح أن OTP و2FA جاهزتان من جهة العميل [2] .
- شاشة Onboarding الجذرية (`OnboardingRootScreen`) موجودة كقالب كما يظهر في الـrouter، لكنها تفتقر إلى المراحل التفصيلية (تمهيدية فقط).
- لا توجد حاليًا شاشات `welcome_screen.dart` أو `permissions_screen.dart` أو `preferences_screen.dart` في الكود الحالي. يجب إنشاؤها.

- بالنسبة لـDSR، الكود الحالي يحتوي على شاشات إعدادات DSR (`dsr_export_screen.dart` , `dsr_erasure_screen.dart`) ضمن Settings [4] ، بينما الخطة تقترح إنشاءها ضمن مجلد منفصل.

- **الفجوات والتعارضات:**

- **شاشات Onboarding مفقودة أو غير مكتملة:** يجب إضافة شاشات الترحيب والأذونات والتفضيلات التي لا توجد حاليًا.
- **التوثيق (Auth) جزئي مفعَّل:** شاشات OTP/2FA موجودة لكنها تحتاج تنشيط سير العمل وربط Back-end (منظر السيرفر). لا تعارض لأن التصميم يسمح بذلك.
- **DSR موجودة في إعدادات المستخدم:** بينما الخطة تقترح مجلد `lib/screens/dsr/` جديد، الواقع أن الشاشات موجودة تحت `lib/screens/settings/` [4] .

- التوافق مع معاييرنا: الخطة تحترم منع استيراد SDK المباشر واستخدام الشيمز (DSR تستخدم `dsr_ux_adapter` الشيمزية) [4] [1] .

- **إنجازات سريعة vs عمل ثقيل:**

- **إنجازات سريعة:**
  - ربط شاشات المصادقة الحالية بـ `AppShell` ونمط التصميم الموحد (من المسار A) وإزالة أي حالة تعليق.
  - الاستفادة من شاشات DSR الموجودة داخل `settings` وفقًا لهيكل الشيمز (تعديل بسيط فقط للوصول إليها إذا لزم).
- **عمل ثقيل:**
  - بناء شاشات Onboarding الثلاثة من الصفر (ولاء بنية التنقل بينها).
  - إعادة تنظيم مكان شاشات DSR إن لزم الأمر، وربطها بتدفقات الحالة الحالية.

توصيات التنفيذ

اقتراح ترتيب التنفيذ: يفضّل البدء أولًا بـ Track A (نظام التصميم وAppShell) لضمان أساس واجهة موحّد قبل تغيير بقية الشاشات. ثانيًا، يمكن العمل على **Track D** (Onboarding/Auth/DSR) وذلك لأن اكتمال تدفق تسجيل الدخول والإعداد ضروري قبل استخدام الميزات الأخرى. بعد ذلك يأتي **Track B** (رحلات) باعتبارها تتمحور حول الوظائف التطبيقية الرئيسية التي تعتمد على القالب الجديد والمصادقة الجاهزة. أخيرًا **Track C** (طرود/طعام) توضع في نهاية التسلسل؛ فهي مستقلة نسبيًا (تعتمد فقط على البنية الشيمزية والطعام يكون Flag). يتيح هذا الترتيب رؤية سريعة للنتائج (واجهة موحّدة، ثم وظائف أساسية).

المخاطر والأسئلة المفتوحة:

- مدى ضرورة إطلاق ميزة الطعام (Food) الآن أم فقط توفير شاشة "قريبًا" خلف Feature Flag؟
- ما مدى تكامل باكند الرحلات المطلوب للمراحل الأولى (استعجال الحجز والتتبع)؟ هل نستخدم بيانات مزيفة أوليًا؟
- هل لدينا بنية تحتية لدعم الإشعارات والتتبع المتقدمة فعليًا، أم يلزم توفير Kill-Switch لكل خدمة (كما جرت العادة)؟
- هل قيود الجدول الزمني تفرض تسليم MVP لكل جزء أم التنفيذ الكامل لكل مسار؟
- مستوى التغطية الاختبارية المطلوب: هل 158 اختبار يجب المحافظة عليها دائمًا (بما في ذلك اختبارات الـUI التي قد تنشأ عند إضافة الشاشات)؟

الامتثال لمعاييرنا الداخلية:

- الخطة تحترم تكامل **الشيمز فقط** ولا تستورِد أي مكتبة SDK مباشرة في `app/lib` (تم التشديد على ذلك في القيود) [1].
- الحفاظ على بنية Clean-B (بما يشمل CI/B-Style) مذكور ضمن الشروط الصارمة؛ نحتاج للتأكد من تجاوز الـAnalyzer والصلاحية الثابتة بعد أي تغييرات.
- المحافظة على عدد الاختبارات أو زيادتها: يجب أن تمرّ جميع الاختبارات القديمة (على الأقل كما في الخطة) [2]، وإضافة اختبارات جديدة لكل ميزات معتمدة.
- نظافة الشفرة وجودة الكود (Analyzer صفر، والتزام بقواعد Banned-Imports) مستمرة كما في الحالة الحالية [2][1].
- أخيرًا، مبدأ العمل وفق نظام تصميم موحد (من مسار A) يعزز اتساق الواجهة ويطابق سجل النظافة (Analyzer=0) الحالي [2][3].

يسعى تنفيذ هذه التوصيات إلى ضمان الانتقال السلس من المرحلة الحالية إلى خطة Manus دون الاضطرار لإعادة كتابة واسعة، مع التركيز على الاستفادة من البنية الحالية واستهداف المخاطر المبكرة (مثل تصميم الواجهات وتشغيل الميزات الأساسية) أولًا.

---

[1][2][3][4] PROJECT_STATUS_v3.2.1.md
https://github.com/abdulrhmanasami/DW_workspace/blob/7e29d9226adc4d75d27f8ff013c7a2d790dc21a1/docs/reports/PROJECT_STATUS_v3.2.1.md