uOttawa

SEG2505 – Introduction au génie logiciel – Automne 2022

Projet Android automne 2022 : application Mealer (20 %)

Instructions

1. Vous travaillerez dans la même équipe que celle que vous avez rejoint pour vos laboratoires
2. Vous soumettrez chaque livrable sous forme de version sur GitHub
3. À la fin du semestre, l'équipe doit démontrer une application complète. Par exemple, un membre de l'équipe montrant un écran avec une fonctionnalité tandis qu'un autre membre montrant un autre écran avec une autre fonctionnalité (fonctionnant sur un téléphone différent), **NE SERA PAS accepté**. L'équipe doit produire une seule application avec toutes les fonctionnalités requises.

Intégrité académique

Tout le travail que vous faites pour satisfaire les exigences de ce cours doit être le vôtre, à moins que la collaboration soit explicitement autorisée. Voir ou copier le travail d'un autre individu (même s'il est stocké dans un répertoire public) ou retirer du matériel d'un livre, d'un magazine, d'un site Web ou d'une autre source - même en partie - et le présenter comme le vôtre constitue une fraude académique, tout comme montrer ou donner votre travail, même partiel, à un autre étudiant.

Description du projet

Vous développez une application de partage de repas basée à Ottawa appelée *« Mealer »* où les cuisiniers locaux peuvent vendre des repas aux clients depuis leur domicile. L'application supporte trois types d'utilisateurs :

- ***Cuisinier***: un utilisateur qui prépare des repas à la maison et les vend aux ***Clients***.
- ***Client***: un utilisateur qui achète des repas du ***Cuisinier.*** Il commande le repas via l'application et vient le chercher de chez le ***Cuisinier*** .
- ***Administrateur***: un utilisateur qui reçoit des plaintes concernant un ***Cuisinier*** de la part d'un ***Client*** et peut suspendre le ***Cuisinier*** si nécessaire.

Dans les sections suivantes, nous décrirons l'application en détail du point de vue de chaque utilisateur.

Client

Pour devenir ***Client***, un utilisateur doit s'inscrire en soumettant les informations suivantes :

- Prénom
- Nom
- Adresse courriel (qui sert également comme nom d'utilisateur)
- Mot de passe du compte
- Adresse
- Informations de carte de crédit (pour fins de paiement)

Une fois inscrit, un ***Client*** peut rechercher des repas en spécifiant un ou plusieurs des critères suivants :

- Nom de repas
- Type de repas (ex. plat principal, soupe, dessert, etc.)
- Type de cuisine (ex. italienne, chinoise, grecque, etc.)

Les résultats obtenus ne doivent afficher que les repas actuellement proposés par les ***Cuisiniers***. De plus, le résultat pour chaque repas montre les informations sur les ***Cuisiniers*** (nom, adresse et description qu'ils ont écrite sur eux-mêmes) ainsi que les informations du repas (prix, type de repas, type de cuisine, liste des ingrédients, allergènes, et description). Le ***Client*** peut alors sélectionner un repas, spécifier une heure de cueillette et soumettre une demande d'achat du repas. Le ***Client*** peut consulter sa demande d'achat. Chaque demande d'achat peut avoir l'un des trois états suivants :

- En attente : le ***cuisinier*** n'a pas encore traité la demande.
- Approuvé : le ***cuisinier*** a approuvé la demande et donc l'achat est terminé et le ***client*** peut ramasser le repas à l'heure indiquée.
- Rejeté : le ***cuisinier*** a rejeté la demande et l'achat n'est donc pas complété.

Un ***Client*** peut consulter la liste des repas qu'il a acheté. A partir de la liste, il peut appuyer sur un repas pour évaluer son ***Cuisinier*** et/ou déposer une plainte. La plainte est évaluée par l'***Administrateur*** qui peut soit la rejeter ou suspendre le ***Cuisinier***.

Administrateur

L'***Administrateur*** est un utilisateur pré-enregistré. Cela signifie que les informations sur le compte de l'***Administrateur*** (c'est-à-dire le nom d'utilisateur et le mot de passe) sont déjà dans la base de données lorsque le système est lancé pour la première fois.

L'**A*dministrateur*** possède une boîte de réception pour recevoir les plaintes des ***Clients***. Chaque plainte est associée à un ***Cuisinier*** et contient une description de la plainte telle que rédigée par le ***Client***. Selon la plainte, ils ont le choix de suspendre le ***Cuisinier*** temporairement ou indéfiniment. Ils peuvent également rejeter la plainte. Une fois que l'**A*dministrateur*** traite la plainte (rejette ou suspend), la plainte disparaît de sa liste.

Cuisinier

Pour devenir ***Cuisinier***, un utilisateur doit s'inscrire en fournissant les informations suivantes :

- Prénom
- Nom

- Adresse courriel (qui sert également comme nom d'utilisateur)
- Mot de passe du compte
- Photo d'un chèque annulé (afin d'être payé)
- Adresse (à partir de laquelle le ***Client*** peut ramasser le repas)
- Brève description de soi-même

Une fois inscrit, le ***Cuisinier*** peut ajouter des repas à son *menu*. Pour ajouter un repas au *menu*, il doit préciser les informations suivantes :

- Nom du repas
- Type de repas (ex. plat principal, soupe, dessert, etc.)
- Type de cuisine (ex. italienne, chinoise, grecque, etc.)
- Liste des ingrédients
- Allergènes
- Prix (le prix qu'il souhaite facturer au ***Client*** pour le repas)
- Description (autres informations que le ***Cuisinier*** souhaite ajouter, tel que le nombre de personnes que le repas peut servir)

Le ***Cuisinier*** peut également supprimer un repas du *menu*.

Un repas ajouté au *menu* ne signifie pas nécessairement qu'il est actuellement offert par le ***Cuisinier***. Cela signifie simplement qu'il peut être offert par le ***Cuisinier***.

Pour offrir un repas, le ***Cuisinier*** doit sélectionner le repas dans son *menu* et confirmer qu'il souhaite l'offrir. Ensuite, le repas est ajouté à la liste des repas offerts par le ***Cuisinier*** et commencera à apparaître dans les recherches du ***Client***. Le ***Cuisinier*** peut supprimer le repas de la *liste des repas proposés*, ce qui signifie que le ***Client*** ne peut plus le voir dans ses recherches ou l'acheter.

Un ***Cuisinier*** possède une liste des demandes d'achat. La liste contient les demandes d'achat soumises par les ***Clients***. Un ***Cuisinier*** peut voir les demandes d'achat et soit les approuver ou les rejeter. Une demande d'achat précise le repas, le nom du ***Client*** et l'heure de cueillette.

Un ***Cuisinier*** peut consulter son profil qui contient les données indiquées lors de son inscription ainsi que les informations suivantes :

- Nombre de repas vendus
- Évaluation

La note d'évaluation est un nombre compris entre 1 et 5 et correspond à la note moyenne reçue des ***Clients***.

Conception d'interface utilisateur

Ce cours ne se concentre pas sur la conception d'interface utilisateur. Cependant, les étudiants sont encouragés à produire des interfaces utilisateur esthétiques. Tenez compte des directives de conception Android lors de la conception de votre application : https://developer.android.com/design.

Livrables

Le projet est divisé en 4 livrables incrémentaux. Les étudiants sont tenus de soumettre chaque livrable avant la date limite.

Livrable	Date d'échéance
1 – GitHub et comptes d'Utilisateurs (3 %)	21 octobre
2 – La plupart des fonctionnalités de l'Administrateur (3 %)	4 novembre
3 – La plupart des fonctionnalités du Cuisinier (3 %)	18 novembre
4 – La plupart des fonctionnalités du Client (ainsi que l'intégration) (9%)	7 décembre
5 – Démo (2%)	Dernière semaine de cours

Le projet doit être réalisé tout au long de la session et les étudiants sont **fortement** encouragés à garder un journal de leurs activités de projet car ces informations seront nécessaires pour le rapport final.

Votre application doit être écrite en Java et construite à l'aide d'Android Studio (vous pouvez utiliser un autre IDE, mais les AEs ne peuvent supporter qu'Android Studio). À la fin du semestre, vous devez implémenter et soumettre une application qui fonctionne basée sur les spécifications. Firebase ou SQLite peuvent être utilisés pour stocker et récupérer les données d'application.

Livrable 1 - GitHub et Comptes d'Utilisateurs

(Lisez attentivement la section « Description du projet » avant de commencer à travailler sur ce livrable.)

Vous devez rejoindre notre classe GitHub, créer votre équipe et accepter ce projet. Veuillez vous référer à "Étapes pour créer votre équipe sur GitHub" pour des instructions détaillées.

Pour ce livrable, vous devez implémenter les composantes de gestion de compte pour les ***Clients*** et les ***Cuisiniers.*** L'application doit permettre aux ***Clients*** et ***Cuisiniers*** de s'inscrire pour des comptes d'utilisateurs. Les **A*dministrateurs*** n'ont pas besoin de s'inscrire car ils sont préinscrits.

Une fois que l'utilisateur s'est connecté, il devrait voir un deuxième écran avec le message suivant : "Bienvenue !" Vous êtes connecté en tant que « rôle » (où « rôle » peut être ***Cuisinier***, ***Client*** ou ***Administrateur***). L'utilisateur peut également se déconnecter. Aucune fonctionnalité supplémentaire ne doit être implémentée à ce stade.

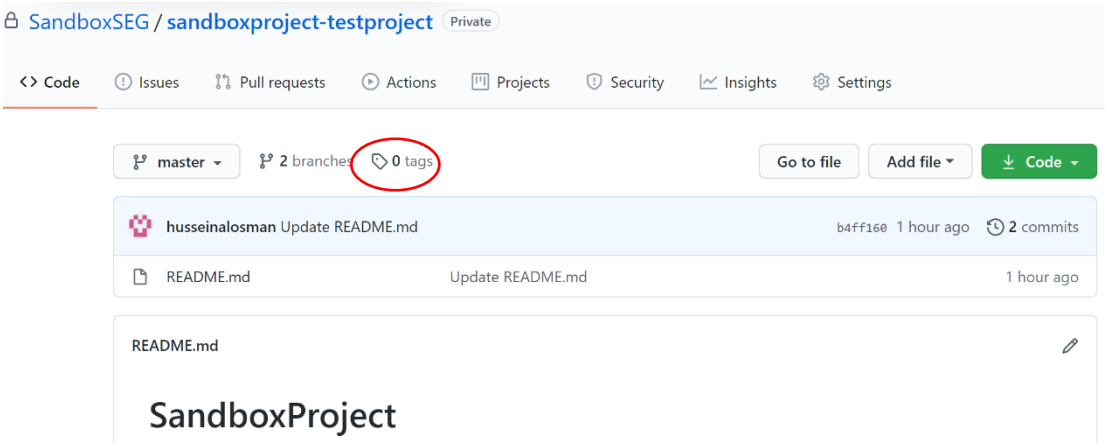
Vous pouvez utiliser Firebase ou SQLite comme base de données. Si vous n'utilisez pas de base de données à ce stade, vous pouvez simplement stocker les informations en mémoire (mais elles seront perdues lorsque vous fermerez l'application) ou dans un fichier.

Dans votre référentiel GitHub, incluez un document PDF contenant un diagramme de classes UML de votre modèle de domaine. Cela n'inclura que les classes UML liées au livrable 1. De plus, dans votre référentiel, fournissez un fichier README contenant les informations d'identification nécessaires pour se connecter en tant *qu'**Administrateur***.

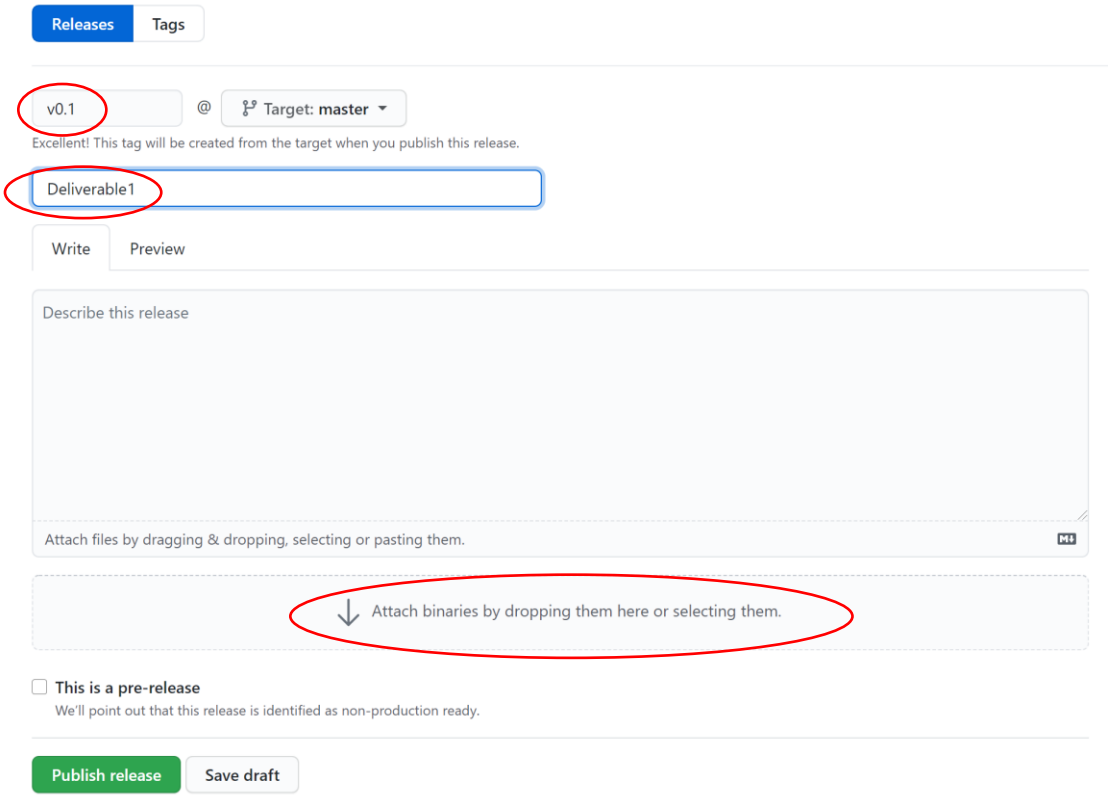
Instructions de Soumission

Pour soumettre votre code, vous allez créer une version à partir de votre référentiel comme suit :

1. Dans votre référentiel, cliquez sur "tag" (voir figure ci-dessous)



2. Cliquez sur le bouton "Create new release" et remplissez le formulaire comme suit (voir figure ci-dessous)
    a. Pour "Tag version" entrez v0.1
    b. Pour "Release title" entrez Livrable1
    c. A partir de la section du formulaire où vous pouvez joindre des fichiers binaires, téléchargez l'APK de votre application
    d. Assurez-vous de renommer l'APK après le nom de votre équipe "groupe1_debug_apk".



Livrable 1 – Schéma d'évaluation

Fonction ou Tâche	**% Pondération (sur 100)**
L'équipe créée dans la salle de classe GitHub contient tous les membres du groupe	10
Chaque membre du groupe a effectué au moins une soumission dans le référentiel.	20
Diagramme de classes UML de votre modèle de domaine : ● (-2 pour chaque classe manquante) ● (-2 pour chaque généralisation incorrecte) ● (-0,5 pour chaque multiplicité incorrecte) ● (-0,5 pour chaque attribut manquant)	25
L'APK est soumis.	5
Un utilisateur peut créer un compte ***Client*** ou ***Cuisinier***.	10
L'***Administrateur***, le ***Cuisinier*** ou le ***Client*** peut voir « l'écran d'accueil » après une authentification réussie. L'écran d'accueil précise le rôle de l'utilisateur.	10
L'utilisateur peut se déconnecter.	10
Tous les champs sont validés. Il existe des messages d'erreur appropriés pour les entrées incorrectes.	10

(-1 pour chaque champ dans lequel l'entrée de l'utilisateur n'est pas validée)	
Optionnel – L'équipe utilise une base de données (ex. Firebase, SQLITE ou autre technologie similaire)	**+5** (bonus)

Livrable 2 – La plupart des fonctionnalités de l'Administrateur

(Lisez attentivement la section « Description du projet » avant de commencer à travailler sur ce livrable.)

Pour ce livrable, vous devez implémenter les fonctionnalités de l'***Administrateur***. De plus, si vous ne l'avez pas encore fait, vous devriez commencer à utiliser une base de données (par exemple, Firebase, SQLITE ou autre technologie similaire).

L'application doit permettre à l'**A*dministrateur*** de voir une liste de plaintes. Pour chaque plainte, l'***Administrateur*** peut soit rejeter la plainte, soit suspendre le ***cuisinier*** associé à la plainte temporairement ou indéfiniment. Une fois que l'***Administrateur*** traite la plainte (rejette ou suspend), la plainte disparaît de la liste des plaintes.

Les plaintes ne doivent pas être codées en dur (hard coded). Elles doivent être stockées dans une base de données. Cependant, étant donné que les **C*lients*** ne peuvent pas encore acheter de repas et soumettre des plaintes à ce stade, vous allez pré-créer une liste de plaintes stockées dans la base de données. Chaque plainte doit être associée à un ***Cuisinier*** (vous avez déjà implémenté la fonctionnalité d'enregistrement pour le livrable 1).

Si un ***Administrateur*** suspend un ***Cuisinier*** en résultat d'une plainte, lorsque le ***Cuisinier*** se connecte, il devrait voir un message l'informant que son compte est suspendu et quand la suspension sera levée en cas de suspension temporaire.

Vous devez inclure au moins **4 cas de tests unitaires** (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).

Dans votre référentiel GitHub, incluez un document PDF contenant un diagramme de classes UML de votre modèle de domaine. Cela n'inclura que les classes UML liées aux livrables 1 et 2. De plus, dans votre référentiel, fournissez un fichier README contenant les informations d'identification nécessaires pour se connecter en tant *qu'**Administrateur***.

Instructions de Soumission

Pour soumettre votre code, créez une version v0.2 dans votre référentiel. Assurez-vous que le fichier APK est ajouté à votre version.

Livrable 2 – Schéma d'évaluation

Fonction ou Tâche	**% Pondération (sur 100)**
Diagramme de classes UML mis à jour de votre modèle de domaine ● (-2 pour chaque classe manquante) ● (-2 pour chaque généralisation incorrecte)	15

● (-0,5 pour chaque multiplicité incorrecte) ● (-0,5 pour chaque attribut manquant)	
L'APK est soumis.	5
Vous avez implémenté 4 cas de test unitaires (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).	20
Lorsqu'un utilisateur s'inscrit, les informations de son compte sont stockées dans la base de données.	10
L'***Administrateur*** peut visionner la liste des plaintes.	10
L'***Administrateur*** peut actionner la plainte (rejeter la plainte ou suspendre le ***Cuisinier***). Pour une suspension temporaire, l'***Administrateur*** peut indiquer la date à laquelle la suspension peut être levée. La plainte disparaît de la liste de l'***Administrateur*** une fois qu'elle est traitée.	15
La liste des plaintes est stockée dans la base de données.	15
Le ***Cuisinier*** peut voir un message lui informant qu'il a été suspendu. Pour une suspension temporaire, le message lui informe quand la suspension sera levée. Pour une suspension permanente, le message lui informe qu'il ne peut plus utiliser l'application.	10
Optionnel – Le groupe fait l'intégration avec CircleCI pour voir les constructions et exécution des unités de test automatisées.	+5 (bonus)

Livrable 3 – La plupart des fonctionnalités du Cuisinier

(Lisez attentivement la section « Description du projet » avant de commencer à travailler sur ce livrable.)

Pour ce livrable, vous implémenterez la majorité des fonctions du ***Cuisinier****.*

Le ***Cuisinier*** doit être en mesure d'ajouter/supprimer des repas de son *menu*. Il doit pouvoir ajouter/supprimer des repas dans son *menu* à la *liste des repas proposés*. Un ***Cuisinier*** suspendu ne devrait pas être en mesure d'effectuer aucune de ces actions précédentes.

Vous devez inclure au moins **4 cas de tests unitaires supplémentaires** (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).

Dans votre référentiel GitHub, incluez un document PDF contenant un diagramme de classes UML de votre modèle de domaine. Cela n'inclura que les classes UML liées aux livrables 1, 2 et 3. De plus, dans votre référentiel, fournissez un fichier README contenant les informations d'identification nécessaires pour se connecter en tant *qu'****Administrateur***.

Instructions de Soumission

Pour soumettre votre code, créez une version v0.3 dans votre référentiel. Assurez-vous que le fichier APK est ajouté à votre version.

Livrable 3 – Schéma d'évaluation

Fonction ou Tâche	**% Pondération (sur 100)**
Diagramme de classes UML mis à jour de votre modèle de domaine ● (-2 pour chaque classe manquante) ● (-2 pour chaque généralisation incorrecte) ● (-0,5 pour chaque multiplicité incorrecte) ● (-0,5 pour chaque attribut manquant)	15
L'APK est soumis.	5
Vous avez implémenté 4 cas de test unitaires (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).	20
Le ***Cuisinier*** peut ajouter un repas au *menu*.	10
Le ***Cuisinier*** peut ajouter un repas à la *liste des repas proposés*.	10
Le ***Cuisinier*** peut supprimer un repas du *menu*. Le ***Cuisinier*** ne peut pas supprimer un repas du *menu* s'il est actuellement dans la *liste des repas proposés*.	10
Le ***Cuisinier*** peut supprimer un repas de la *liste des repas proposés*.	10
Lorsqu'un ***Cuisinier*** suspendu se connecte, il ne peut voir que le message de suspension et se déconnecter. Il ne peut effectuer aucune autre action.	10
Tous les champs sont validés. Il existe des messages d'erreur appropriés pour les entrées incorrectes. (-1 pour chaque champ dans lequel l'entrée de l'utilisateur n'est pas validée)	10

Livrable 4 – La plupart des fonctionnalités du Client (ainsi que l'intégration)

(Lisez attentivement la section « Description du projet » avant de commencer à travailler sur ce livrable.)

Pour ce livrable, vous implémenterez le reste des fonctionnalités de l'application.

Le ***Client*** doit pouvoir rechercher des repas en spécifiant le nom du repas, le type de repas et/ou le type de cuisine. Le résultat de la recherche doit afficher les repas actuellement proposés par des ***Cuisiniers*** non-suspendus**.** Pour chaque repas, le ***Client*** doit pouvoir voir les informations sur le ***Cuisinier*** incluant leur note d'évaluation ainsi que les informations sur le repas. Le ***Client*** peut ensuite soumettre une demande d'achat pour un repas sélectionné à partir des résultats de la recherche.

Le ***Client*** doit être en mesure de voir le statut de leur demande d'achat. En raison de simplicité, une fois qu'une demande d'achat est approuvée par le ***Cuisinier***, nous supposons que le prix du repas est débité de la carte de crédit du ***Client*** (laquelle a été saisie lors de l'inscription) et que le processus d'achat est terminé. Vous n'implémenterez aucune fonctionnalité liée au débit de la carte de crédit.

Le ***Client*** doit pouvoir évaluer les ***Cuisiniers*** auprès desquels il a acheté les repas. De plus, le ***Client*** peut déposer une plainte contre un ***Cuisinier*** auprès duquel ils ont acheté un repas.

Le ***Cuisinier*** doit être en mesure de visualiser et d'approuver/rejeter les demandes d'achat reçues des ***Clients***. Le ***Cuisinier*** doit pouvoir consulter son profil qui contient ses informations et sa note d'évaluation.

Vous devez inclure au moins **4 cas de tests unitaires supplémentaires** (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).

Dans votre référentiel GitHub, incluez un document PDF contenant votre rapport final (voir le schéma d'évaluation pour plus d'informations sur ce qu'il faut inclure dans le document). De plus, dans votre référentiel, fournissez un fichier README contenant les informations d'identification nécessaires pour vous connecter en tant *qu'**Administrateur***.

Instructions de Soumission

Pour soumettre votre code, créez une version v0.4 dans votre référentiel. Assurez-vous que le fichier APK est ajouté à votre version.

Livrable 4 – Schéma d'évaluation

Fonction ou Tâche	**% Pondération (sur 100)**
Diagramme de classes UML mis à jour de votre modèle de domaine ● (-2 pour chaque classe manquante) ● (-2 pour chaque généralisation incorrecte) ● (-0,5 pour chaque multiplicité incorrecte) ● (-0,5 pour chaque attribut manquant)	10
L'APK est soumis.	5
Vous avez implémenté 4 cas de tests unitaires (tests locaux simples). Il n'est pas nécessaire d'inclure l'instrumentation ou les tests Espresso (UI).	10
Rapport final comprenant : ● Page titre (2,5 points) ● Courte introduction (2,5 points) ● Diagramme de classes UML mis à jour ● Tableau précisant les contributions des membres de l'équipe pour chaque livrable (10 points) ● Toutes les captures d'écran de votre application (10 points) ● Leçons apprises (5 points)	30
Le ***Client*** peut rechercher un repas. Le ***Client*** peut voir les résultats de recherche de repas offerts par des ***Cuisiniers*** non-suspendus.	5
Le ***Client*** peut voir les informations sur le ***Cuisinier*** incluant leur note d'évaluation pour chaque repas dans le résultat de la recherche. Le ***Client*** peut voir les informations du repas pour chaque repas.	5
Le ***Client*** peut soumettre une demande d'achat.	5

Le ***Cuisinier*** peut recevoir la demande d'achat soumise par le ***Client***.	
Le ***Client*** peut voir le statut de son achat (en attente, approuvé ou rejeté).	5
Le ***Client*** peut évaluer le ***Cuisinier*** auprès duquel il a acheté un repas.	5
Le ***Client*** peut déposer une plainte concernant un ***Cuisinier*** auprès de l'***Administrateur***.	5
Le ***Cuisinier*** peut consulter et approuver/rejeter les demandes d'achat reçues des ***Clients***.	5
Le ***Cuisinier*** peut voir son profil et sa note d'évaluation.	5
Tous les champs doivent être validés. Il devrait y avoir des messages d'erreur appropriés pour les entrées incorrectes. (-1 pour chaque champ dans lequel l'entrée de l'utilisateur n'est pas validée).	5
Optionnel – Le ***Client*** reçoit une notification lorsque sa demande d'achat est approuvée/rejetée.	+10 (bonus)