Do what you like. Let AI do the rest.

Software Engineer Senior Python (AI) (F/H)

Test Technique.

Contact:

Adresse: 1 Pl. Giovanni da Verrazzano, 69009 Lyon
Téléphone: 04 72 19 61 65
Email: contact@digitalkin.ai

Contents

1 Introduction

Digitalkin, startup innovante spécialisée dans les systèmes multi-agents, est à la recherche d'un(e) **Software Engineer Python (AI) (F/H)** pour rejoindre notre équipe technique basée à Lyon (9ème).

Votre rôle sera de contribuer à la construction de notre architecture agentique tout en réalisant l'intégration d'agents d'intelligence artificielle via des frameworks existants. Vous travaillerez sur des projets novateurs au croisement de l'IA et des systèmes multi-agents, en collaboration avec des experts backend, front-end, et DevOps.

Dans le cadre du processus de recrutement, nous vous proposons le test technique suivant. Ce test est conçu pour refléter les responsabilités et les attentes décrites dans la description du poste, et pour vous donner l'opportunité de démontrer vos compétences techniques, votre créativité et votre capacité à résoudre des problématiques complexes en intelligence artificielle.

2 Proposition de Test Technique

2.1 Objectif

Développer une application Python qui simule une interaction entre un utilisateur et un système de support technique composé de deux agents d'IA collaboratifs. L'objectif est de créer un programme où l'Agent A communique avec l'utilisateur et, en cas de besoin, consulte l'Agent B pour obtenir des informations techniques approfondies afin de fournir des solutions pertinentes.

2.2 Description de la Tâche

Créer une application en Python qui permet :

- À l'utilisateur de poser des questions ou de décrire des problèmes techniques concernant un logiciel.
- À l'Agent A de répondre à l'utilisateur en langage naturel.
- À l'Agent A de consulter l'Agent B lorsque la question dépasse ses connaissances, afin d'obtenir des informations détaillées ou des étapes de résolution.

Vous êtes libre de choisir le framework ou les bibliothèques que vous souhaitez utiliser. Vous pouvez utiliser l'API d'OpenAI (nous fournirons la clé API si nécessaire) ou tout autre service/outil approprié. Il n'est pas attendu de développer des modèles d'apprentissage profond à partir de zéro.

2.3 Exigences

- **Langage** : Utilisez **Python**.
- **Interfaces** : L'application peut être une application console, une application web simple ou toute autre interface que vous jugez appropriée.
- **Agents** :
  - **Agent A** : Interagit directement avec l'utilisateur.

  - **Agent B** : Possède une base de connaissances techniques avancée.
  - **Collaboration** : L'Agent A consulte l'Agent B lorsque nécessaire.
- **Fonctionnalités** :
  - Compréhension et génération de langage naturel pour interagir avec l'utilisateur.
  - Mécanisme pour l'Agent A de décider quand consulter l'Agent B.
  - Transmission efficace de l'information entre les agents.
- **Documentation** : Fournissez des instructions claires sur la façon d'exécuter l'application, ainsi que toute configuration nécessaire.

2.4 Fonctionnalités à Implémenter

1. **Interaction Utilisateur-Agent A** :
   - L'utilisateur peut saisir des questions ou des problèmes techniques.
   - L'Agent A répond en langage naturel.
2. **Communication entre Agent A et Agent B** :
   - L'Agent A analyse la question de l'utilisateur.
   - Si l'Agent A ne connaît pas la réponse, il consulte l'Agent B.
   - L'Agent B fournit des informations détaillées ou des étapes de résolution.
   - L'Agent A transmet la réponse appropriée à l'utilisateur.
3. **Mécanisme de Décision de l'Agent A** :
   - Implémentez une logique pour que l'Agent A décide quand consulter l'Agent B (par exemple, en fonction de mots-clés, de la complexité de la question, etc.).

2.5 Améliorations Optionnelles (si le temps le permet)

- **Interface Utilisateur Améliorée** :
  - Développer une interface graphique simple (par exemple, en utilisant streamlit, chainlit, etc.).
  - Améliorer l'expérience utilisateur avec des éléments visuels.
- **Journalisation et Historique des Conversations** :
  - Enregistrer les conversations pour un suivi ultérieur.
  - Permettre à l'utilisateur de revoir l'historique des interactions.

2.6 Instructions de Soumission

- **Référentiel de Code** :
  - Téléchargez votre code sur un référentiel Git public (GitHub, GitLab, etc.).
  - Incluez un fichier `README` avec :
    * Une brève description de votre application.
    * Des instructions sur la façon d'installer et d'exécuter l'application localement.
    * Toutes les notes ou hypothèses que vous avez faites lors du développement.
- **Structure du Projet** :
  - Organisez votre code de manière claire et logique.
  - Utilisez des messages de commit significatifs si applicable.

2.7 Critères d'Évaluation

- **Fonctionnalité** :
  - L'application répond aux exigences spécifiées.
  - Les agents interagissent correctement entre eux et avec l'utilisateur.
- **Qualité du Code** :
  - Code propre, lisible et maintenable.
  - Bonne organisation du projet et utilisation appropriée des concepts Python.
  - Gestion efficace des exceptions et des erreurs.
- **Utilisation des Technologies d'IA** :
  - Intégration efficace avec des API ou des frameworks IA existants (par exemple, OpenAI API, LangChain, etc.).
  - Compréhension de base du traitement du langage naturel.
- **Créativité et Résolution de Problèmes** :
  - Approche innovante pour la communication entre les agents.
  - Toutes les fonctionnalités ou améliorations supplémentaires ajoutées.
- **Documentation** :
  - Instructions claires pour l'installation et l'exécution.
  - Commentaires utiles dans le code si nécessaire.

2.8 Temps Alloué

Le test est conçu pour être complété en **environ 2 heures**. Il est entendu que toutes les améliorations optionnelles peuvent ne pas être réalisées dans ce laps de temps.

2.9 Objectif de ce Test

Ce test est conçu pour refléter les responsabilités et les attentes décrites dans la description du poste :

- **Maîtrise de Python** : Démontrer votre capacité à développer des applications en Python de manière efficace.
- **Intégration de Solutions d'IA** : Montrer comment vous utilisez des frameworks ou des API d'IA existants pour construire des solutions innovantes.
- **Conception de Systèmes Multi-Agents** : Fournir un aperçu de la façon dont vous abordez la communication et la collaboration entre différents agents.
- **Autonomie et Créativité** : Vous encourager à prendre vos propres décisions concernant l'architecture et l'implémentation, reflétant l'autonomie valorisée dans notre équipe.

2.10 Notes Supplémentaires pour les Candidats

- **Concentrez-vous sur la Qualité plutôt que sur la Quantité** : Il est préférable d'avoir une application bien conçue avec les fonctionnalités de base plutôt qu'une application incomplète avec de nombreuses fonctionnalités.
- **Utilisation de l'API OpenAI** : Nous vous fournirons une clé API OpenAI.
- **Documentez Toute Hypothèse** : Si vous faites des hypothèses pour combler des lacunes dans les exigences, veuillez les documenter dans votre `README`.
- **Questions** : Si quelque chose n'est pas clair, vous êtes les bienvenus pour nous contacter pour des clarifications.

Contact:

Adresse: 1 Pl. Giovanni da Verrazzano, 69009 Lyon
Téléphone: 04 72 19 61 65
Email: contact@digitalkin.ai