Agent.py

```
from typing import List
from crewai import Agent, Crew, Process, Task
from crewai.project import CrewBase, agent, crew, task

@CrewBase
class GameBuilderCrew:
    """GameBuilder crew"""
    agents_config = 'config/agents.yaml'
    tasks_config = 'config/tasks.yaml'

    @agent
    def senior_engineer_agent(self) -> Agent:
        return Agent(
            config=self.agents_config['senior_engineer_agent'],
            allow_delegation=False,
            verbose=True
        )

    @agent
    def qa_engineer_agent(self) -> Agent:
        return Agent(
            config=self.agents_config['qa_engineer_agent'],
            allow_delegation=False,
            verbose=True
        )

    @agent
    def chief_qa_engineer_agent(self) -> Agent:
```

```
    return Agent(
        config=self.agents_config['chief_qa_engineer_agent'],
        allow_delegation=True,
        verbose=True
    )


@task
def code_task(self) -> Task:
    return Task(
        config=self.tasks_config['code_task'],
        agent=self.senior_engineer_agent()
    )

@task
def review_task(self) -> Task:
    return Task(
        config=self.tasks_config['review_task'],
        agent=self.qa_engineer_agent(),
        #### output_json=ResearchRoleRequirements
    )

@task
def evaluate_task(self) -> Task:
    return Task(
        config=self.tasks_config['evaluate_task'],
        agent=self.chief_qa_engineer_agent()
    )
```

```
    @crew
    def crew(self) -> Crew:
        """Creates the GameBuilderCrew"""
        return Crew(
            agents=self.agents,  # Automatically created by the @agent decorator
            tasks=self.tasks,  # Automatically created by the @task decorator
            process=Process.sequential,
            verbose=True,
        )
```

Main.py

```
import sys
import yaml
import os
import subprocess

# The `game_builder_crew` package is optional in this workspace. Import
# defensively so `main.py` can be executed even if the package isn't
# installed (which otherwise raises ModuleNotFoundError on import).
try:
    from game_builder_crew.crew import GameBuilderCrew  # type: ignore
    _HAS_GAME_BUILDER_CREW = True
except Exception:
    GameBuilderCrew = None  # type: ignore
    _HAS_GAME_BUILDER_CREW = False
    # Try local fallback: `agent.py` may define `GameBuilderCrew` in this repo.
    try:
        from agent import GameBuilderCrew as LocalGameBuilderCrew  # type: ignore
```

```
        GameBuilderCrew = LocalGameBuilderCrew  # type: ignore
        _HAS_GAME_BUILDER_CREW = True
    except Exception:
        pass


def run():
    # Replace with your inputs; it will interpolate any tasks and agents information
    if not _HAS_GAME_BUILDER_CREW:
        print('game_builder_crew is not installed in this environment.')
        print("Install it or run a local game directly, for example:")
        print("  python space_crew_game.py")
        return

    print("## Welcome to the Game Crew")
    print('-------------------------------')

    # Try to load examples from package config path, then repo root.
    candidates = [
        os.path.join('src', 'game_builder_crew', 'config', 'gamedesign.yaml'),
        'gamedesign.yaml',
    ]
    examples = None
    for path in candidates:
        if os.path.exists(path):
            try:
                with open(path, 'r', encoding='utf-8') as file:
                    examples = yaml.safe_load(file)
            except Exception as e:
                print('Failed to read', path, e)
```

```
        break

if examples and 'example3_snake' in examples:
    inputs = {'game': examples['example3_snake']}
else:
    # No example file found; prompt user for a game description.
    print('No gamedesign.yaml found or example missing.')
    print('Enter your game prompt, finish with a single line containing only END:')
    lines = []
    try:
        while True:
            line = input()
            if line.strip() == 'END':
                break
            lines.append(line)
    except KeyboardInterrupt:
        print('\nInterrupted. Exiting.')
        return
    prompt = '\n'.join(lines).strip()
    if not prompt:
        print('No prompt entered. Exiting.')
        return
    inputs = {'game': prompt}

try:
    game = GameBuilderCrew().crew().kickoff(inputs=inputs)
except Exception as e:
    print('Error running crew kickoff:', e)
    # Save prompt to a file so user can run it later via run_crew_from_file.py
```

```
        try:
            from datetime import datetime
            ts = datetime.utcnow().strftime('%Y%m%d_%H%M%S')
            pending = f'pending_prompt_{ts}.txt'
            with open(pending, 'w', encoding='utf-8') as f:
                f.write(inputs['game'])
            print(f'Prompt saved to {pending}. When ready, run: python
run_crew_from_file.py {pending} --output generated.py')
            # Offer to generate a simple local fallback game now
            yn = input('Would you like to generate a small local fallback game now? [y/N]:
').strip().lower()
            if yn == 'y':
                out = f'generated_fallback_{ts}.py'
                _fallback_generate_star_game(inputs['game'], out=out)
                print('Wrote fallback game to', out)
        except Exception:
            pass
        return


    print("\n\n############################")
    print("## Here is the result")
    print("############################\n")
    print("final code for the game:")
    print(game)
    after the program execution it generated the game using the prompt
the game file as below


stars.py
import random
import sys
```

```
def make_game(size=8, stars=10, seed=None):
    if seed is not None:
        random.seed(seed)
    player = [size//2, size//2]
    stars_pos = set()
    while len(stars_pos) < stars:
        stars_pos.add((random.randrange(size), random.randrange(size)))
    return size, player, stars_pos

def display(size, player, stars):
    for y in range(size):
        row = []
        for x in range(size):
            if (x, y) == tuple(player):
                row.append('P')
            elif (x, y) in stars:
                row.append('*')
            else:
                row.append('.')
        print(' '.join(row))

def main():
    size, player, stars = make_game()
    print('Star Collector - collect all stars (*)')
    while True:
        display(size, player, stars)
        if not stars:
            print('You collected all stars!')
```

```
            break
        cmd = input('Move (W/A/S/D) or Q to quit: ').strip().lower()
        if not cmd:
            continue
        if cmd[0] == 'q':
            print('Goodbye')
            break
        x, y = player
        if cmd[0] == 'w':
            y = max(0, y-1)
        elif cmd[0] == 's':
            y = min(size-1, y+1)
        elif cmd[0] == 'a':
            x = max(0, x-1)
        elif cmd[0] == 'd':
            x = min(size-1, x+1)
        player[0], player[1] = x, y
        stars.discard((x, y))

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted. Bye!')
#!/usr/bin/env python3
"""Generate a small terminal game locally (fallback generator).

Example:
    python generate_now.py -p "Collect stars on an 8x8 grid" -o mygame.py
```

```
Use `--run` to open the generated script in a new terminal window (Windows, macOS, Linux where supported).
"""
from __future__ import annotations

import argparse
import sys
import os
import subprocess
import tempfile

def main(argv=None) -> int:
    p = argparse.ArgumentParser(description='Generate a small terminal game locally')
    p.add_argument('-p', '--prompt', required=True, help='Game prompt (string)')
    p.add_argument('-o', '--output', help='Write generated game to this file')
    p.add_argument('--run', action='store_true', help='Open the generated script in a new terminal window and run it')
    args = p.parse_args(argv)
    try:
        # Local fallback generator (self-contained) so it doesn't depend on `main.py`.
        def _fallback_generate_star_game(prompt: str, out: str | None = None) -> str:
            code ='''#!/usr/bin/env python3

        Tiny Star Collector

Controls: W/A/S/D to move. Collect all stars (*) on an 8x8 grid.'''

import random
```

```
import sys

def make_game(size=8, stars=10, seed=None):
    if seed is not None:
        random.seed(seed)
    player = [size//2, size//2]
    stars_pos = set()
    while len(stars_pos) < stars:
        stars_pos.add((random.randrange(size), random.randrange(size)))
    return size, player, stars_pos

def display(size, player, stars):
    for y in range(size):
        row = []
        for x in range(size):
            if (x, y) == tuple(player):
                row.append('P')
            elif (x, y) in stars:
                row.append('*')
            else:
                row.append('.')
        print(' '.join(row))

def main():
    size, player, stars = make_game()
    print('Star Collector - collect all stars (*)')
    while True:
        display(size, player, stars)
        if not stars:
```

```
            print('You collected all stars!')
            break
        cmd = input('Move (W/A/S/D) or Q to quit: ').strip().lower()
        if not cmd:
            continue
        if cmd[0] == 'q':
            print('Goodbye')
            break
        x, y = player
        if cmd[0] == 'w':
            y = max(0, y-1)
        elif cmd[0] == 's':
            y = min(size-1, y+1)
        elif cmd[0] == 'a':
            x = max(0, x-1)
        elif cmd[0] == 'd':
            x = min(size-1, x+1)
        player[0], player[1] = x, y
        stars.discard((x, y))

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('\\nInterrupted. Bye!')
```

output:

```
Star Collector - collect all stars (*)
```

. . . . . .

. * . * * . . .

. * . . . . . .

. * * . . . . .

. . . * P . . .

. * . . . . . .

. . . . . * .

. . . . . *

Move (W/A/S/D) or Q to quit:w

. . . . . .

. * . * * . . .

. * . . . . . .

. * * . P . . .

. . . * . . . .

. * . . . . . .

. . . . . * .

. . . . . *

Move (W/A/S/D) or Q to quit:a

. . . . . .

. * . * * . . .

. * . . . . . .

. * * P . . . .

. . . * . . . .

. * . . . . . .

. . . . . * .

. . . . . *

Move (W/A/S/D) or Q to quit:a

. . . . . .

. * . * * . . .

. * . . . . . .

. * P . . . . .

. . . * . . . .

. * . . . . . .

. . . . . * .

. . . . *

Move (W/A/S/D) or Q to quit:q

Goodbye

After colleted all the stars it display the you have collected all the stars