BRIDGE

Key Highlights:

- AI powered, Multi-agent conversational system design.
- Purpose: To help students understand role specific responsibilities and simulate an interprofessional collaboration in healthcare education.
- Parallel Processing for optimised and faster responses.
- Efficient User Interface.

Currently available agents:

Nurse: Patient care, empathy, and practical nursing observations
Physician Assistant: Medical diagnostics, clinical decision-making, and treatment planning
Medical Social Worker: Social care, support services, and community resources
Physical Therapist: Patient rehabilitation, physical therapy techniques, and recovery plans
Public Health Professional: Population health, disease prevention, health education, and community well-being
Health Administrator: Organizational management, resource coordination, policy implementation, and interdisciplinary team support

Input format

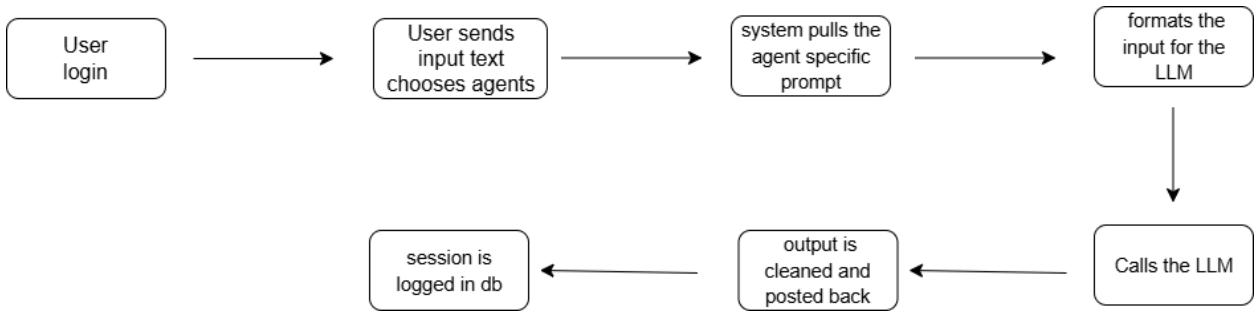
```
{
  "case_study": "Your medical case study text here...",
  "content": "Your question or prompt for the agents",
  "agent_ids": ["agent1", "agent2", "..."],
  "perspective": "Optional perspective context"
}
```

Sample Output format

```
{
  "responses": {
```

```
    "nurse": {
      "id": "response_id_123",
      "text": "As a nurse caring for Julia..."
    },
    "physician": {
      "id": "response_id_456",
      "text": "From a clinical perspective..."
    },
    "socialworker": {
      "id": "response_id_789",
      "text": "Considering Julia's psychosocial needs..."
    }
  }
}
```


High level Workflow



Database:

Contains 2 tables

Users

to store user profiles

```python
class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    supabase_user_id = Column(UUID(as_uuid=True), unique=True, nullable=False, index=True)
    full_name = Column(Text, nullable=False)
    age_range = Column(Text, nullable=False)
    professional_role = Column(Text, nullable=False)
    years_in_practice = Column(Integer, nullable=False)
    academic_degrees = Column(Text, nullable=False)
    certifications = Column(Text)
    specialties = Column(Text)
    opioid_experience = Column(Text)
    created_at = Column(DateTime(timezone=True), server_default=func.now())
```

Chatsession

To store chat logs

```python
class ChatSession(Base):
    __tablename__ = "chat_sessions"

    session_id = Column(UUID(as_uuid=False), primary_key=True, index=True)  # Fixed: Use UUID type but store as string
    response_ids = Column(JSONB)
    memory = Column(Text)  # Use Text instead of String for large content
    created_at = Column(DateTime(timezone=True), server_default=func.now())
    updated_at = Column(DateTime(timezone=True), onupdate=func.now())
```

Tech stack

Models currently supported : OpenAI - gpt-4o-mini
AWS services used:
EC2 : hosting the app
API gateways: for secure routing
S3: for storage

Backend tech: FastAPI, PostgresSQL
Frontend: React, Tailwind CSS and TypeScript

Folder Structure

IPE
|-- \frontend
(all front files and supabase files)
|-- \src
|------ \agents
|------architecture.py (Agent architecture & centralized resource manager)
|------ base_agent.py (base class def)
|------ prompt.py (agents prompts and general instructions to LLM)
|------ memory.py (memory class to add and summarize a message)
|------ \appserver
|------ db.py (Database configuration and access management)
|------ models.py (DB table classes)
|------ routes.py (API routes using the resource manager)
|------ \core
|------background_tasks.py (background processing for summarization)
|------ config.py (configuration management)
|------ health.py (api health check utility)
|------ resource_manager.py (resource manager for backend services)
|------ \rag
|------ \test
|------api.ipynb
|------ url.yaml
|------ logger_utils.py
|-- Deployment.md
|-- Readme.md
|-- License
|-- config.env.template (config files)
|-- gunicorn.conf.py (config files)
|-- requirement.txt
|-- start_dev.sh (Development startup script)
|-- start_server.sh (Production startup script)
|-- student_sim.sh
|-- test_performance.py (tests BRIDGE’s API payload capacity and the responses)