**System Architecture**

The Nurse Scheduling System is designed to efficiently allocate shifts to nurses across multiple days, ensuring optimal coverage while considering constraints such as minimum staffing requirements. The system leverages a Harmony Search algorithm for optimization and is built using Flask for the web interface, SQLAlchemy for database management, and a combination of HTML, CSS, and JavaScript for the frontend.

**Architecture Diagram**

+------------------------+

**User Interface**

(HTML, CSS, JavaScript)

+------------------------+

**Flask Application**

- Routes (app/routes.py)

- Harmony Search (app/harmony\_search.py)

- Models (app/models.py)

+------------------------+

**Database Layer**

- SQLite Database

- SQLAlchemy ORM

+------------------------+

**Components**

1. **User Interface**

**Description**: Provides an interactive web interface where users can view and interact with the nurse schedules.

**Technologies**: HTML, CSS, JavaScript.

**Files**:

* + - templates/index.html: Renders the schedule in a user-friendly format.
    - static/style.css: Defines the styling for the web pages.

1. **Flask Application**

**Description**: Acts as the server-side logic that processes requests, runs the optimization algorithm, and interacts with the database.

**Files**:

* + - app/routes.py: Defines the routes and handles HTTP requests. It initializes the Harmony Search algorithm and manages the scheduling process.
    - app/harmony\_search.py: Contains the implementation of the Harmony Search optimization algorithm, which generates and evaluates nurse schedules.
    - app/models.py: Manages the interaction with the SQLite database, including table creation and data insertion.

1. **Database Layer**

**Description**: Stores the data related to nurses, shifts, and schedules. SQLAlchemy ORM is used to manage database interactions.

**Technologies**: SQLite, SQLAlchemy.

**Files**:

* + - Database schema definitions and data storage are managed through SQLAlchemy, which is initialized and configured in app/models.py.

**Flow of Operations**

1. **User Interaction**:
   * Users access the web interface through a browser. The frontend renders the schedule using the data provided by the Flask application.
2. **Request Handling**:
   * When a user visits the main route (/), the Flask application processes the request by initializing the Harmony Search algorithm with the nurse scheduling configuration.
3. **Optimization**:
   * The Harmony Search algorithm runs to generate an optimal schedule based on constraints and preferences. This involves:
     + Generating a random initial population of schedules.
     + Iteratively improving schedules through improvisation, evaluation, and updating the harmony memory.
4. **Database Operations**:
   * The optimized schedule is saved to the SQLite database using SQLAlchemy. The database schema includes tables for days, shifts, nurses, and schedules.
5. **Schedule Display**:
   * The final schedule is formatted and rendered on the web page, allowing users to view the allocated shifts and nursing staff for each day.

**Technologies Used**

* **Flask**: Web framework for handling requests and serving the application.
* **SQLAlchemy**: ORM for database interactions, providing a high-level abstraction over SQLite.
* **Harmony Search**: Optimization algorithm for generating efficient schedules.
* **HTML/CSS/JavaScript**: Technologies used for creating and styling the user interface.

**Advantages**

* **Scalability**: The system can be easily adapted to include more nurses, shifts, or days.
* **Flexibility**: The use of Harmony Search allows for the accommodation of various constraints and preferences.
* **User-Friendly**: The web interface provides a clear and accessible way for users to view and interact with the schedules.