HADES: Holistic Astronomical Database Exploration System

Client & Advisor: Dr. Goce Trajcevski

By: Andrew Snyder, Alex Polston, Alek Norris,
Eamon Collins, James Byrd, Svyatoslav Varnitskyy

sdmay25-20

Binary Star System

- A pair of stars bound to each other by gravity
- Revolving around a common center of mass within each others orbit
- Provide one of the only ways to measure a stars mass
- Insight into stellar evolution

Project Overview

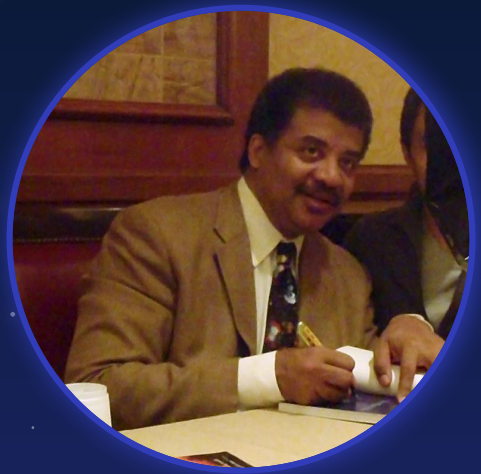
- Objective: Develop an intuitive system to query and display binary star data generated by the POSYDON simulation project
- Key Features:
  - Import multivariable time-series simulation data into relational database
  - Enable custom SQL and natural language queries
  - Provide predefined sample queries
- Deliverables:
  - Web Based Application
  - Relational Database
  - Data Ingestion Tool

Concept

Web app for querying, database connection, and data ingestion

Users

Astrophysicists
Expert



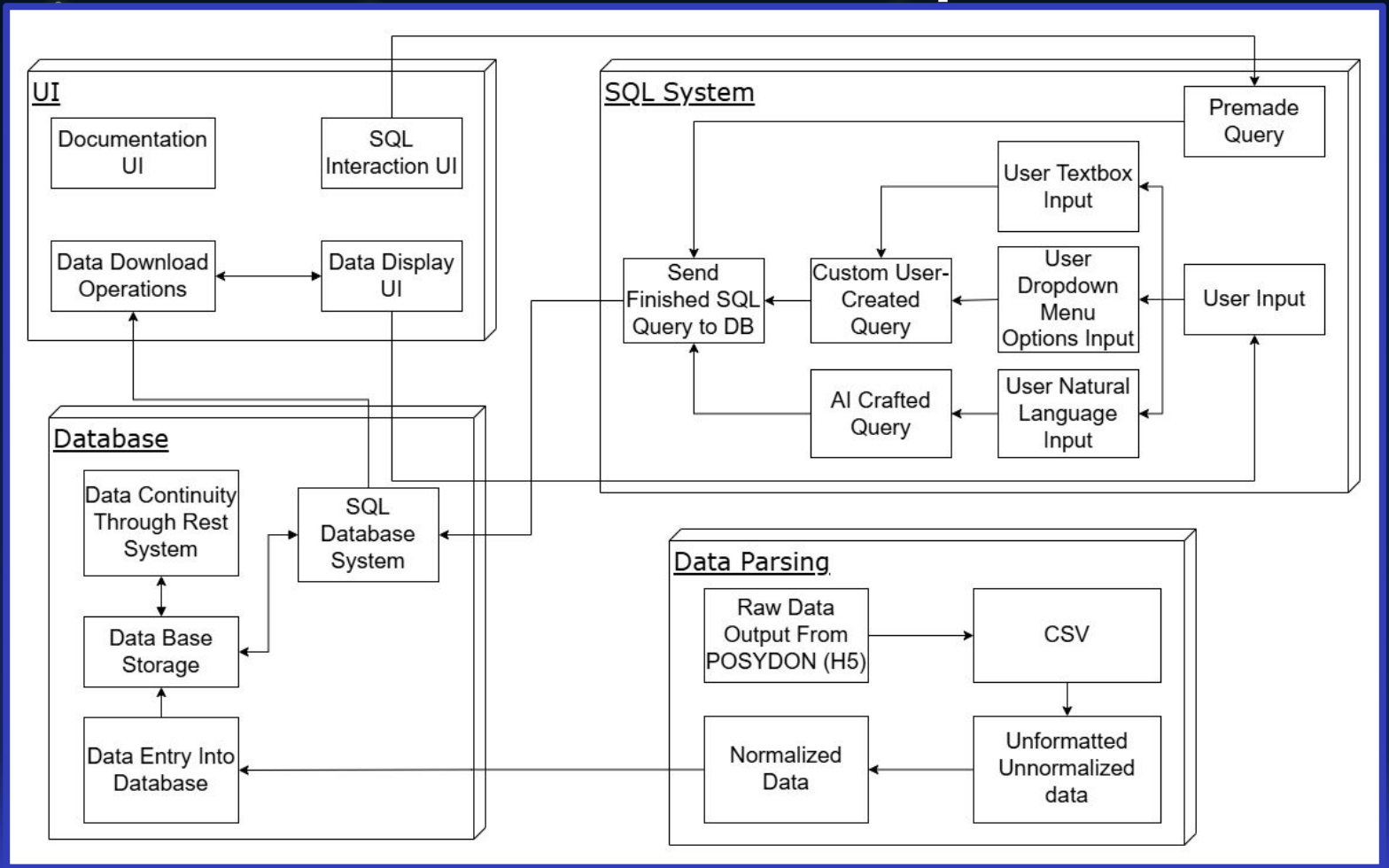
Educators
Intermediate

Students
Novice

Requirements

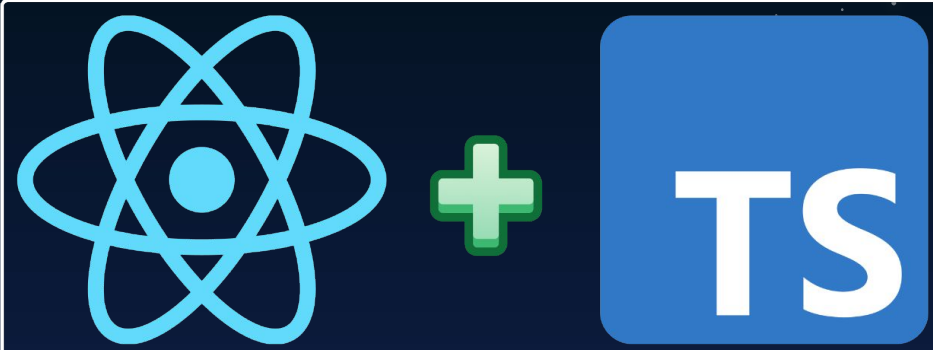
Functional	Non-Functional
• Supports Custom and Built-in queries • Converts natural language into SQL • Saves queries • Converts compressed csv files into database schema	• Scalability • Visually appealing user interface • Time-efficient data parsing • Clear presentation of data

Detailed Design



Tech Stack

Back End - Python

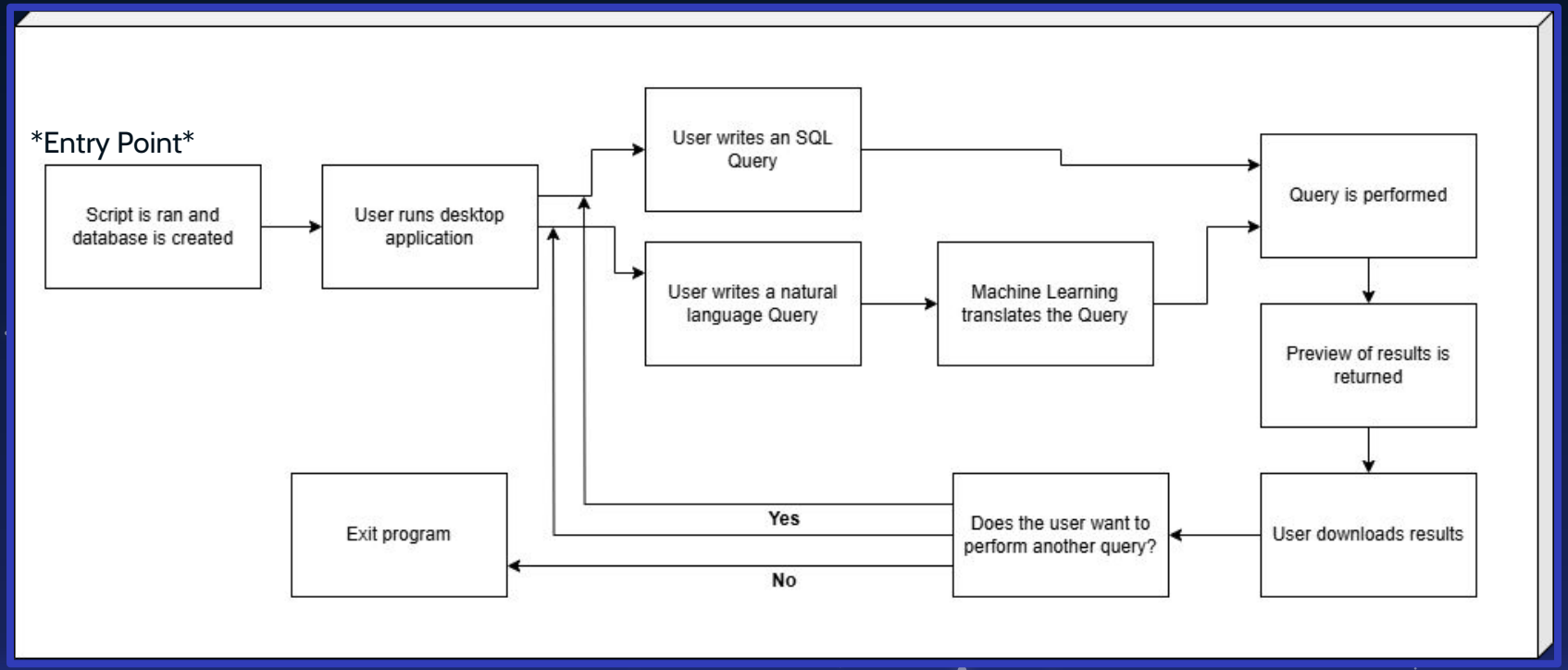


Front End - React + TypeScript

Database - PostgreSQL

LLM - Open AI

User Interaction Pathway



Project Timeline and Management

Task Category	Task	1/20 - 27 1	1/27 - 2/3 2	2/3 - 10 3	2/10 - 17 4	2/17 - 24 5	2/24 - 3/3 6	3/3 - 10 7	3/10 - 17 8	3/17 - 24 9	3/24 - 31 10	3/31 - 4/7 11	4/7- 14 12	4/14 - 21 13	4/21 - 28 14	4/28 - 5/5 15
Research	SQL + Database															
Design Elements	Database Implementation															
Develop Software	Software Implementation															
Develop Software	Database Implementation															
Develop Software	Connection of Database & UI															
Debug	UI and NLP Debug															
Debug	Database Debug															
Documentation	Documentation															

Testing Strategy

- Project features highly dependent subsystems
- Complete automated testing coverage for all code
- Detailed results through manual testing process
- Team reviews of each change

Automated Testing

General
- Must automate unit, integration, and system tests for all new code
- Automation runs on commit to the team Git repository
- Acts as regression testing

Backend
- Unit Testing - Tox
- Integration Testing - Tox
- System Testing - Tox

Frontend
- Unit Testing - Jest
- Integration Testing - Cypress
- System Testing - Cypress

Manual Testing

Interface Testing

- Database: Query with list of sample queries, note results
- Backend: Run requests from team Postman group, note results
- Frontend: Navigate through website, record video

Acceptance Testing

- Continuous testing of functional components
- Collect numerical data on performance
- Discuss visual decisions as a team
- Review changes with client/advisor weekly

NLP SQL ProtoTyping

Prompting

Initial Generated Query / Adjusted Query

nothing showed up, could you show me one, including "logRho" where mass is greater than 1.5?

You can use the following SQL query to look at `logRho`, `mass`, and `energy` values where `mass` is greater than 1.5 from the `"public.grid_0_0142_grid_run0_final_profile1"` table:

sql   Copy code

```sql
SELECT mass, logRho, energy
FROM "grid_0_0142_grid_run0_final_profile1"
WHERE mass > 1.5;
```

Queries and Results

Database Queried

```
hades=# SELECT mass, "logRho", energy FROM "grid_0_0142_grid_run0_fi
nal_profile1" where mass > 1.5;
        mass          |         logRho         |        energy
----------------------+------------------------+-----------------------
-
  2.428273796159032   |   -9.136390283830577   |   1918837430976.8416
 2.4282737961569523   |   -9.136385738294923   |   1918846682484.2263
 2.4282737961536687   |   -9.136376285548819   |    1918865921319.566
  2.428273796145799   |   -9.13635[illegible]  |   [illegible]18912279573.6323
  2.428273796126791   |   -9.136309798045636   |    1919001227357.688
  2.428273796094218   |   -9.136225588910927   |   1919172564990.2378
 2.4282737960273955   |    -9.13605283777916   |   1919523938069.9016
```

No Results

Generated Query

Native Data

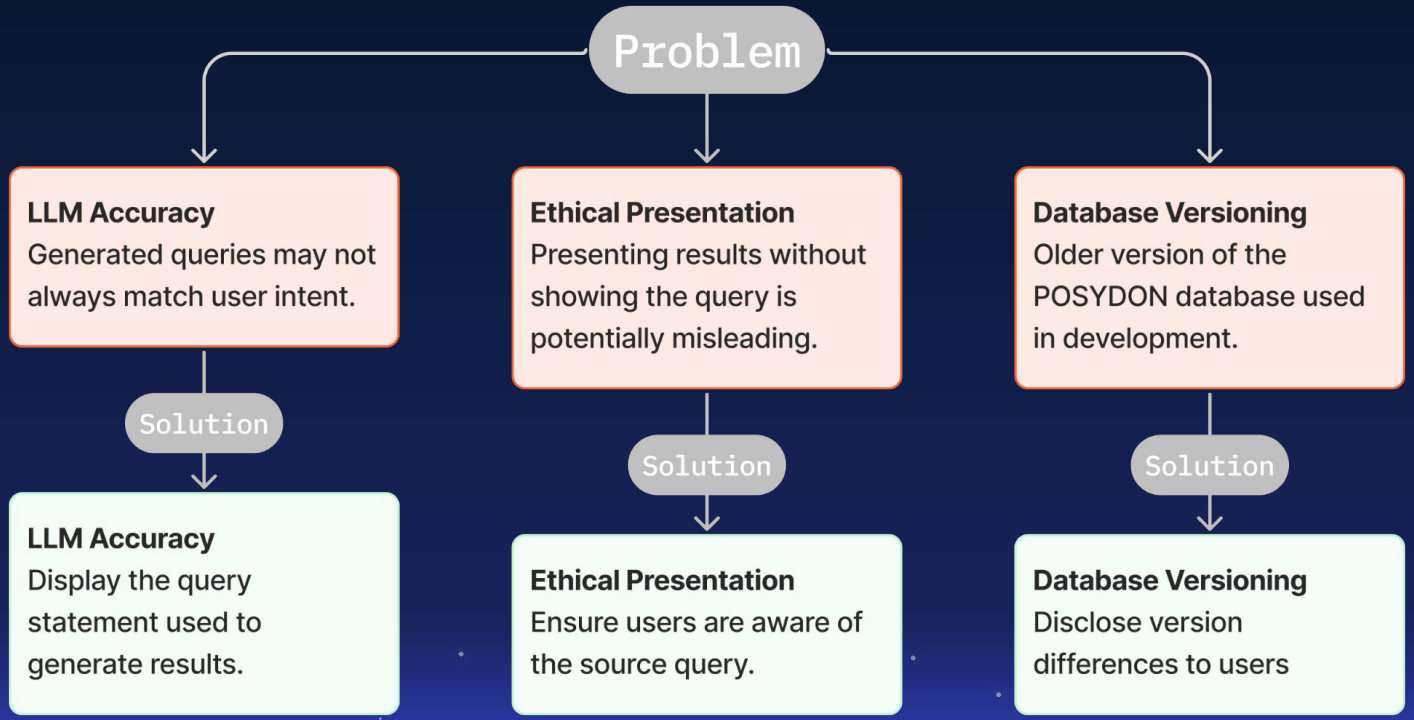
RAW H5 File

MASS < 5

	radius	mass	logRho	omega	energy	x_mass_fraction	y_mass_fract
0	36.602	2.428	-9.136	0.000	18837430976.8	0.000	0.986
1	36.602	2.428	-9.136	0.000	18846682484.2	0.000	0.986
2	36.602	2.428	-9.136	0.000	18865921319.5	0.000	0.986
3	36.602	2.428	-9.136	0.000	18912279573.6	0.000	0.986
4	36.602	2.428	-9.136	0.000	19001227357.6	0.000	0.986
5	36.602	2.428	-9.136	0.000	19172564990.2	0.000	0.986
6	36.602	2.428	-9.136	0.000	19523938069.9	0.000	0.986

Implementation

Ensuring Accuracy and Transparency in Query Generation



Risks & Mitigation

Risks	Mitigation
Missing Deadlines	Strong communication and cooperation on tasks
Software Struggles	Use well-established technologies, be prepared to switch if necessary
Slow Database Querying	Index certain data for quick reference, or partition database. Mostly outside our scope
NLP Malfunctioning	Thorough testing and training

Conclusions

Current State

- Client approves the design and early prototypes
- Prototypes
  - Frontend
  - Database proof of concept
  - Parsing Script
  - GPT Script
- Working front-back interconnection

Future Plans

- Automate the setup process
  - PostgreSQL installation
  - Dependency configuration
  - Database initialization
- Improve Database & UI communication
  - Finish script integration
- Enhanced Functionality
  - Query history & examples
  - GPT training for SQL schema and NLP improvements.
- GPT API Funding

Thank You

Questions?