Detecting convoys of molecular and atomic trajectories

Team Members: Jason Guo, Jeremy Lewis, Daryl Kay, Claira Springer

Iowa State University
Adviser: Goce Trajcevski

Project Vision

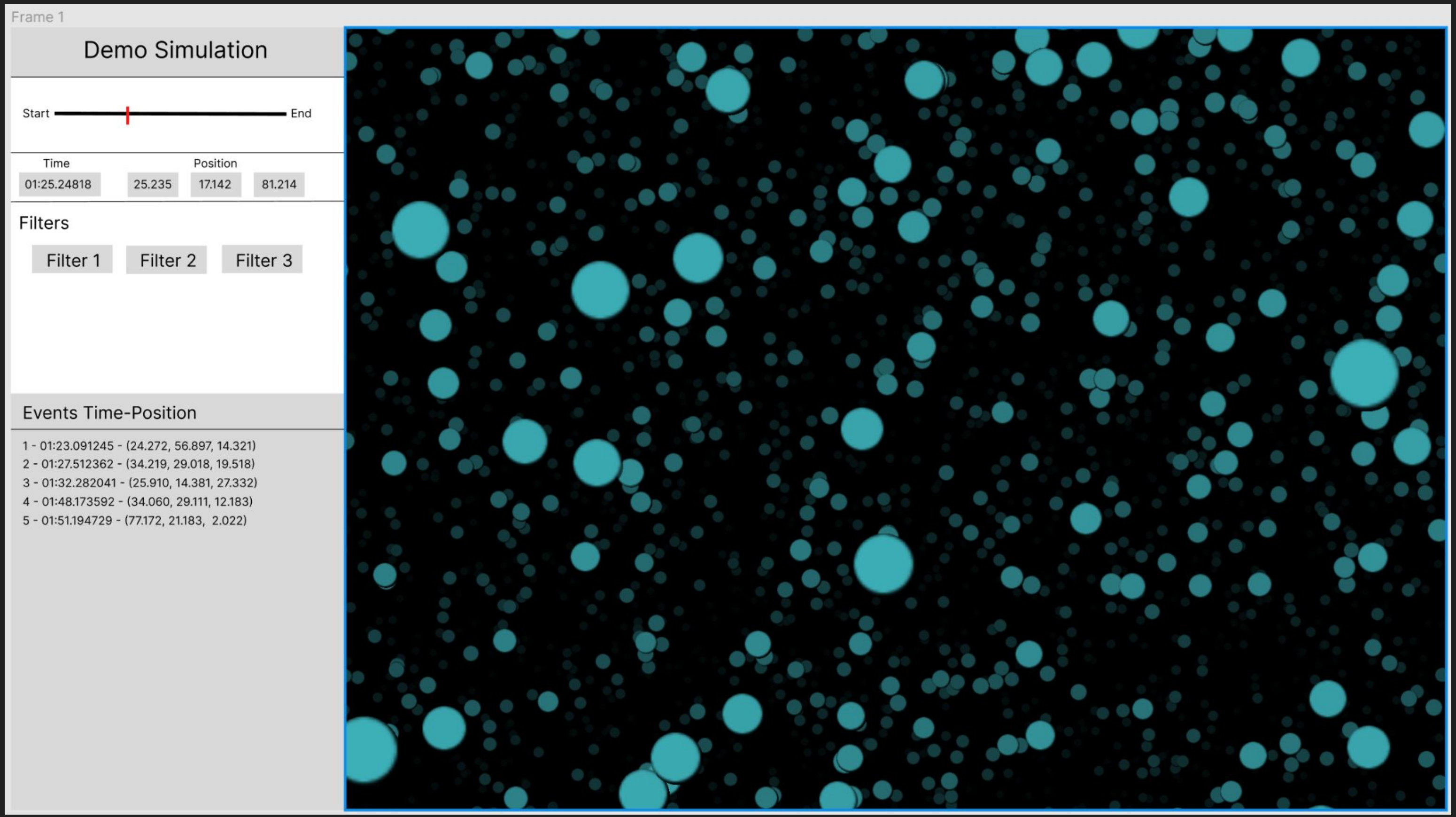
- High cost to testing drug interactions with substrates over time
- Simulations lower these costs of this testing
- Interesting events in simulation data (e.g., Hydrogen bond)
- Develop a system that will detect "interesting events"
- Graphically display results

Who cares?

- Scientists
- Experts from other fields
- Pharmaceutical companies
- Chemists pursuing research/further funding

Conceptual/Visual Sketch

- Front end UI
- Backend
- Scientists
- Find and view targeted interesting events



Demo Simulation

Time	Position		
01:25.24818	25.235	17.142	81.214

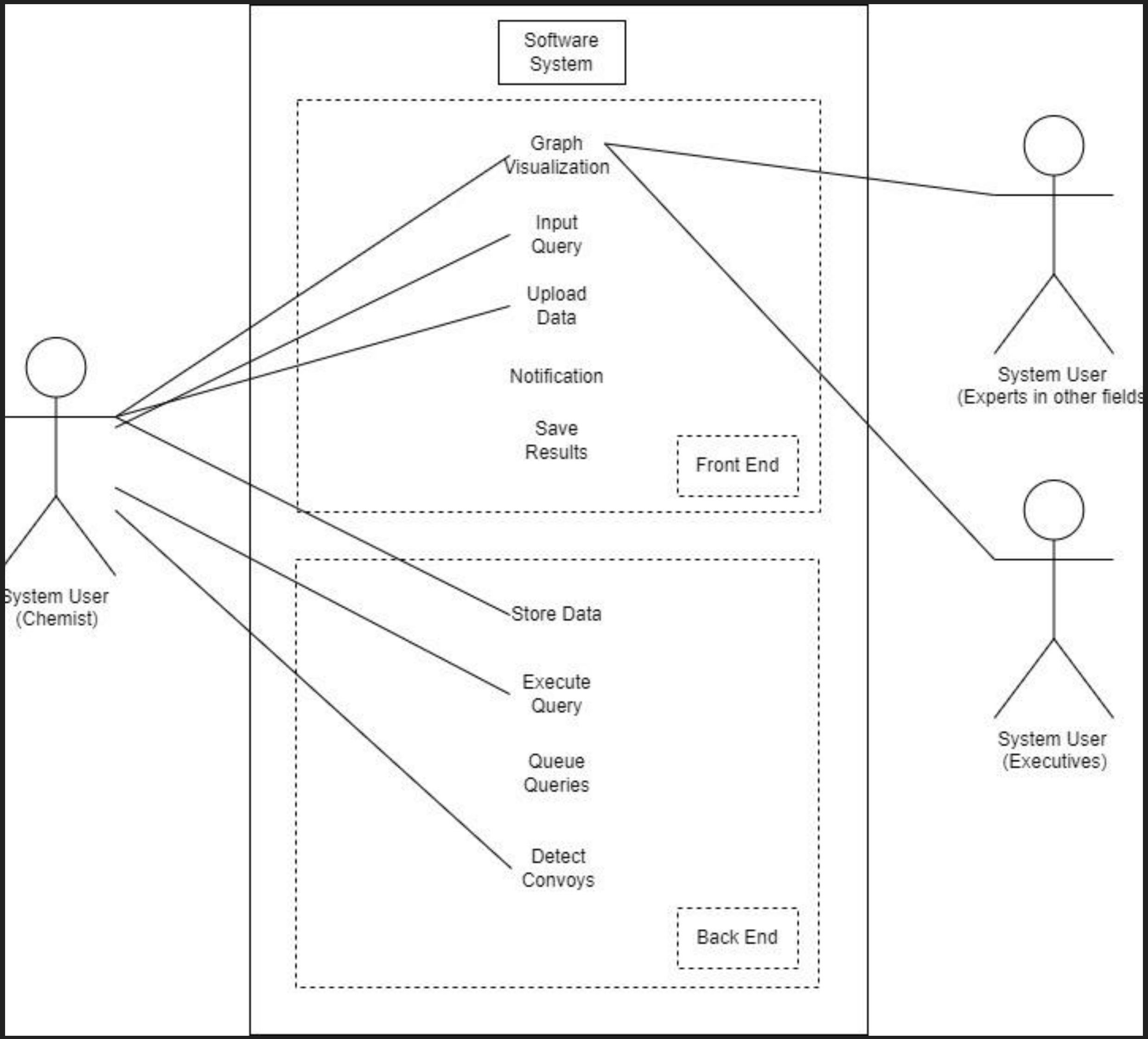
Events Time-Position

1 - 01:23.091245 - (24.272, 56.897, 14.321)
2 - 01:27.512362 - (34.219, 29.018, 19.518)
3 - 01:32.282041 - (25.910, 14.381, 27.332)
4 - 01:48.173592 - (34.060, 29.111, 12.183)
5 - 01:51.194729 - (77.172, 21.183, 2.022)

Requirements

- Functional
  - Scale with users
  - Computations on backend
  - Parameters are verified
- Non-functional
  - Intuitive UI
  - Minimal downtime when uploading data
  - File type conversions
- Technical and/or other constraints
  - Notifications
  - Constant runtime
  - $300 budget

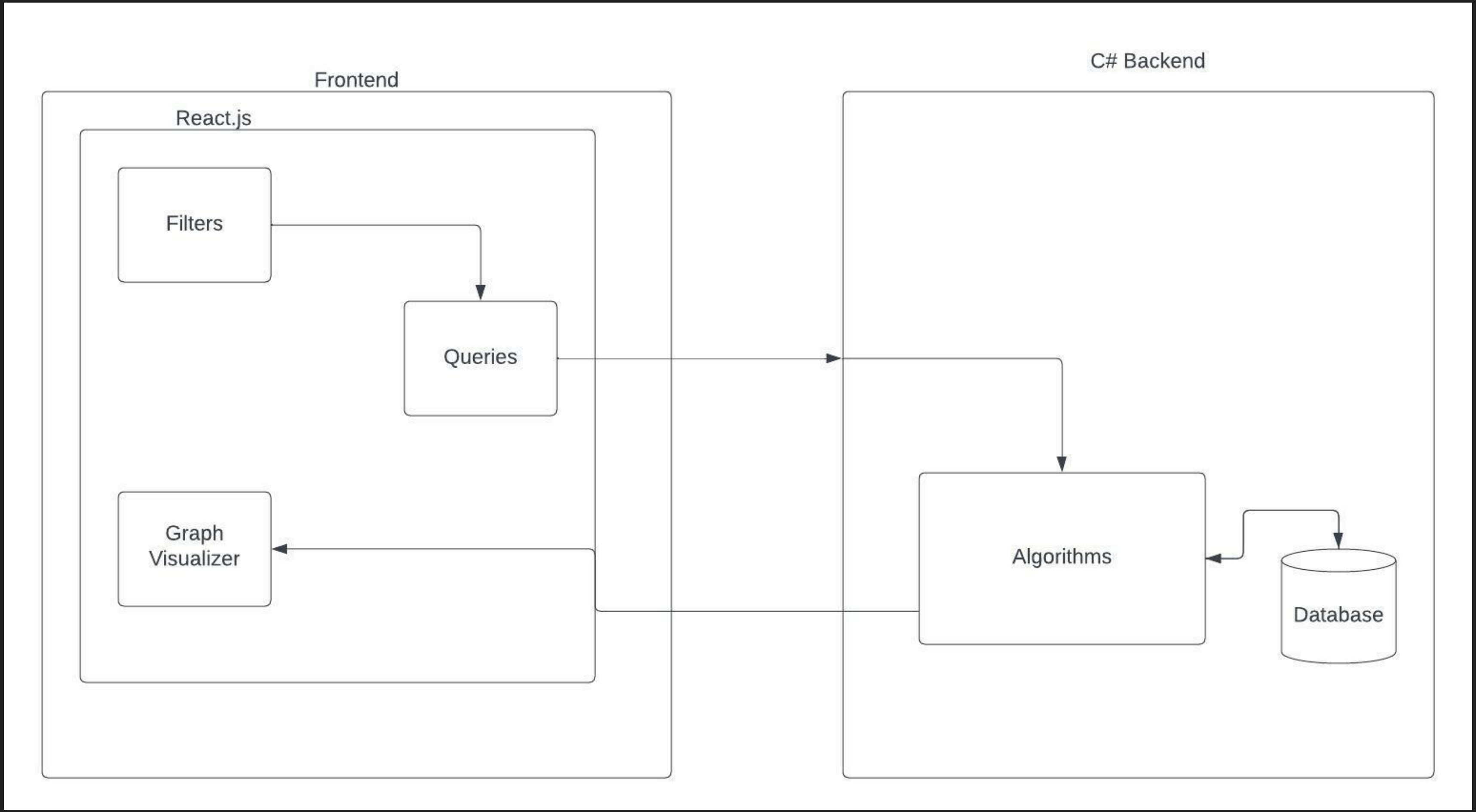
Conceptual Design Diagram



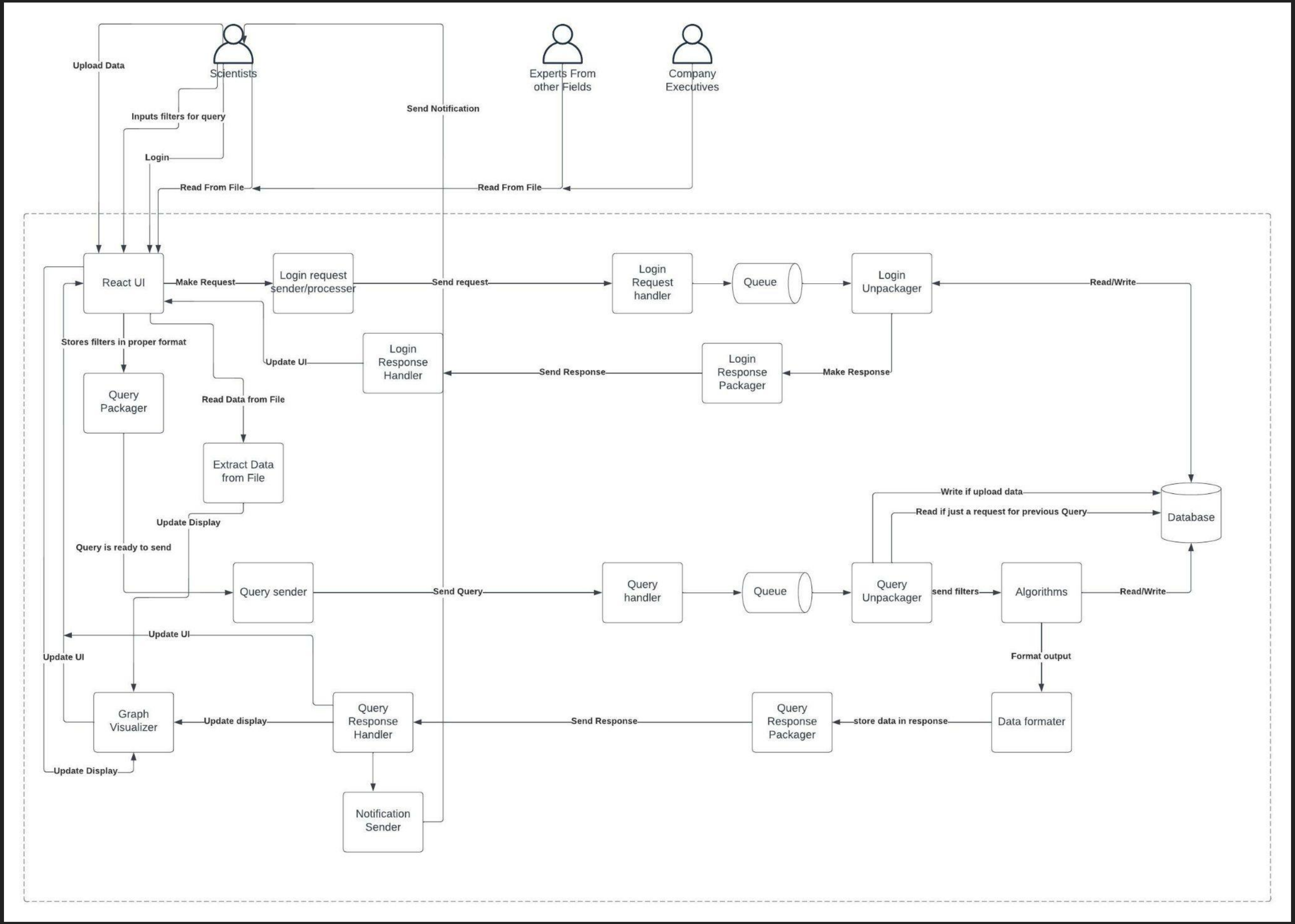
System Design (3-5 slides)

- Functional decomposition/system architecture/software architecture
- Component diagram
- Detailed design (components/modules design), interfaces, UI/UX
- HW/SW platforms, technology, frameworks, standards
- **3-5 slides, 5-6 minutes**

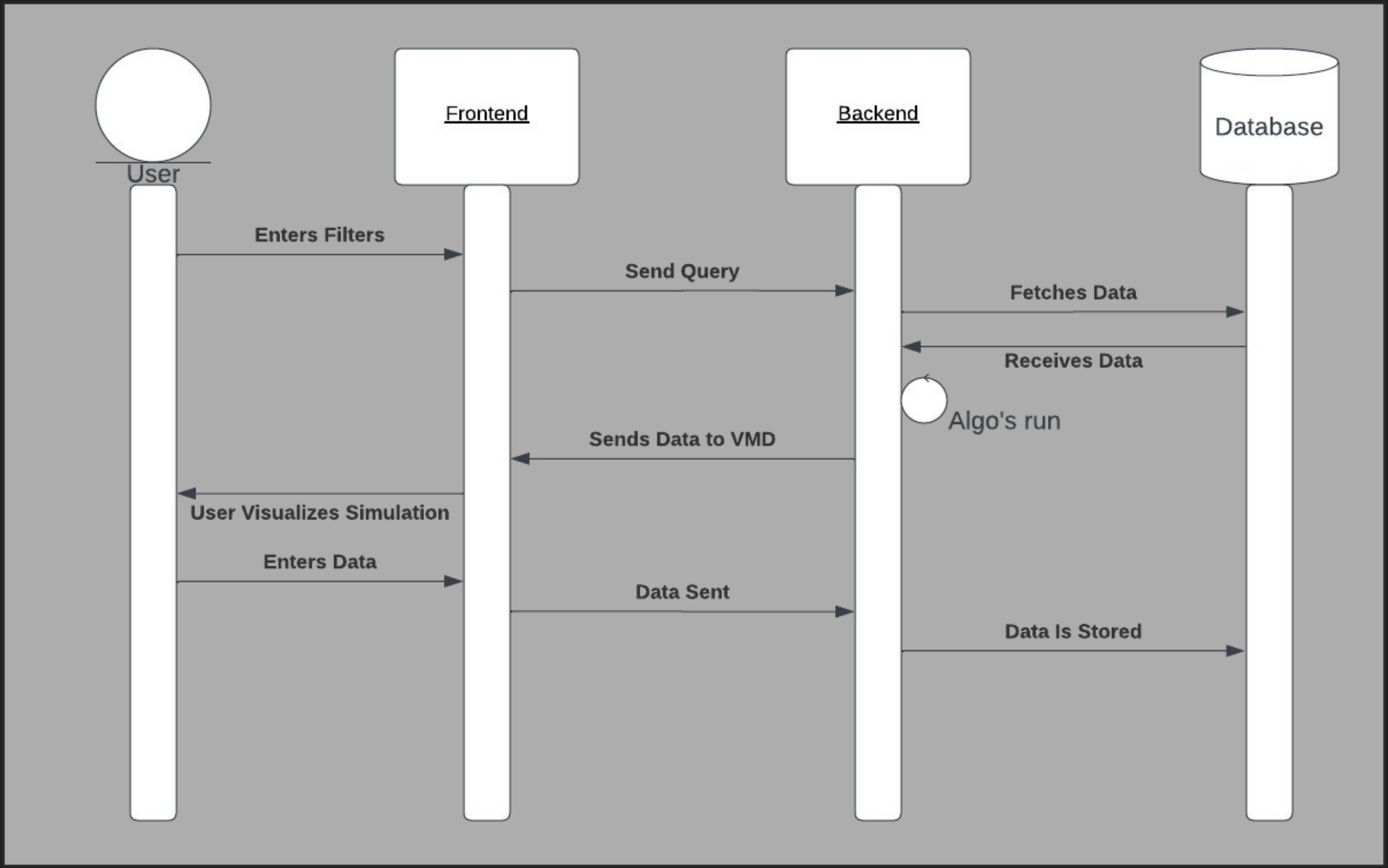
Software Architecture Overview



Component Diagram



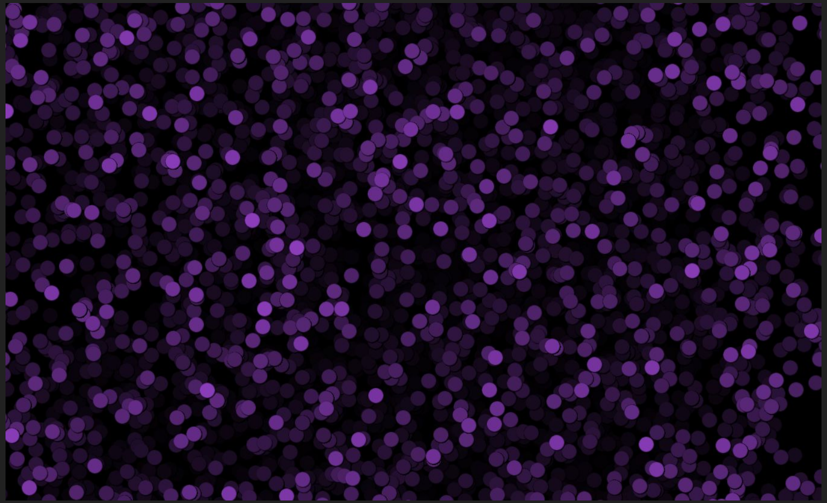
Example Sequence Of Events for Querying



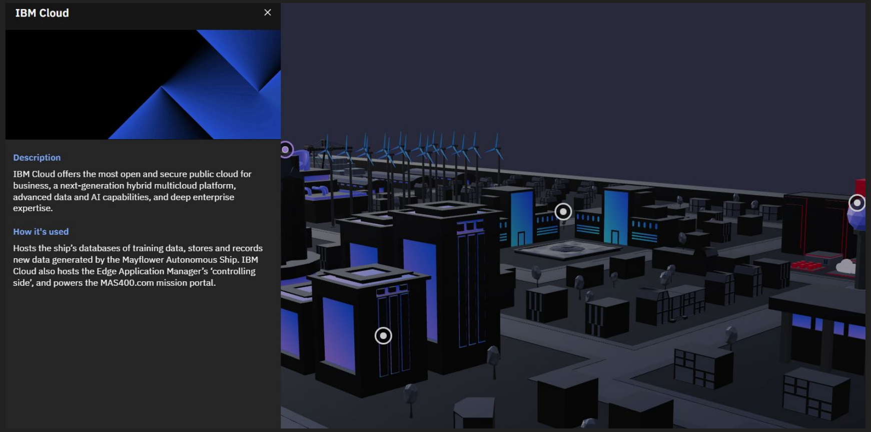
Design Complexity

- Graphing large amounts of data
- Query and perform calculations on that data
- Outputting data to the user in a timely manner (~30 minutes)
- Chemistry - Motions of atoms and groups of molecules

Frontend Graphing Library



Three.js



IBM Cloud

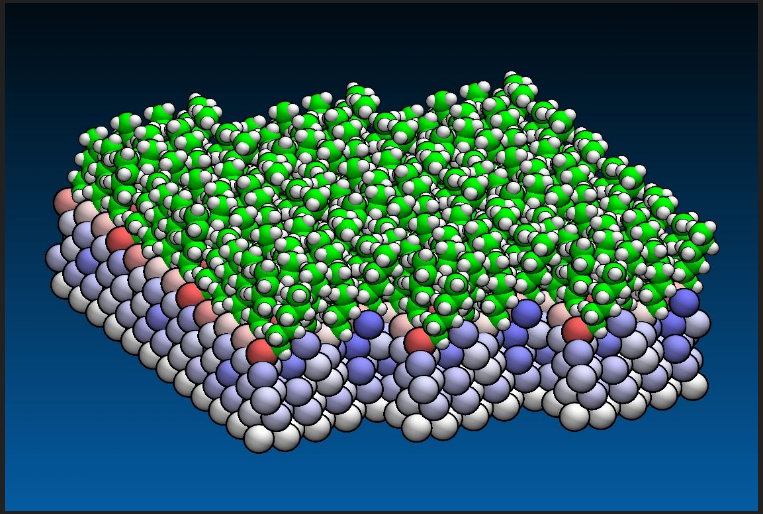
Description

IBM Cloud offers the most open and secure public cloud for business, a next-generation hybrid multicloud platform, advanced data and AI capabilities, and deep enterprise expertise.

How it's used

Hosts the ship's databases of training data, stores and records new data generated by the Mayflower Autonomous Ship. IBM Cloud also hosts the Edge Application Manager's 'controlling side', and powers the MAS400.com mission portal.

Babylon.js



Visual Molecular Dynamics (VMD)

Design Choices

- Front end technology - React.js
  - Graphing library - VMD
- Backend API - C#/.Net
- Database - MySQL

Standards

- IEEE/EIA 12207 - Framework for the software development lifecycle
- IEEE/ISO/IEC 90003-2018 - Software cycle and customer satisfaction standards
- ISO/IEC 27001 - Information security
- V. Phoha, "A standard for software documentation," in Computer, vol. 30, no. 10, pp. 97-98, Oct. 1997, doi: 10.1109/2.625327.

Project Plan - Tasks and Risks

- Tasks:
  - UI
  - Data storage
  - Graphing display
  - Query processing
  - Notifications
  - Testing
  - Backend algorithms implemented

- Risks:
  - Dataset too large to graph effectively
  - Front end graphing library fails
  - Testing difficulties
  - Corrupted data
  - Speed wanted isn’t feasible with the dataset size (Unrealistic expectations)

Project Plan - Risk Mitigation

Risk	Probability	Mitigation
Frontend library fails to function as intended	0.3	Alternative graphing libraries
Data corruption	0.1	Data backup
Server unable to handle/is slow in processing the computations required	0.2	Research and optimization
Testing for accuracy of the system is difficult	0.2	Testing libraries and frameworks
The data set is too large to graph effectively	0.5	Dividing and splitting data

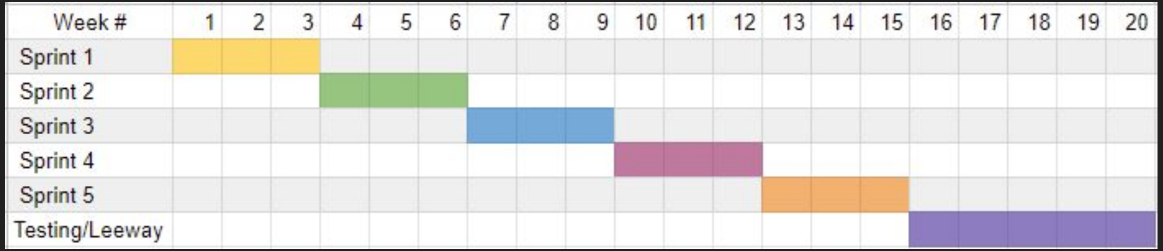
Project Plan – Schedule/Milestones (1-2 slides)

Milestones include:

- Sprint 1/2
  - Having a working graphing library integrated into the frontend that is able to graph spherical objects and have a load time of less than a second
  - Uploading and ability to read/process data from the simulation asynchronously
- Sprint 3
  - Frontend is able to send queries 95% of the time with no issues
  - Able to graph potential data in less than 2 seconds
  - The backend receives and processes queries 95% of the time with no issues
  - The backend has implemented algorithms that run without any issues
- Sprint 4
  - Frontend can read responses from backend and graph results of queries in less than 2 seconds 99% of the time with no issues
  - The backend receives and processes queries 99% of the time with no issues
  - Backend manages queues of queries with no issues
- Sprint 5
  - 90% test coverage on front and back end

Milestones include:

- Sprint 1/2
  - Functional graphing library
  - Asynchronous upload/read/processing of data
- Sprint 3
  - 95% reliability
  - Graphing within 2 seconds
  - Functional Backend
- Sprint 4
  - 99% reliability in Front/Backend
- Sprint 5
  - 90% test coverage

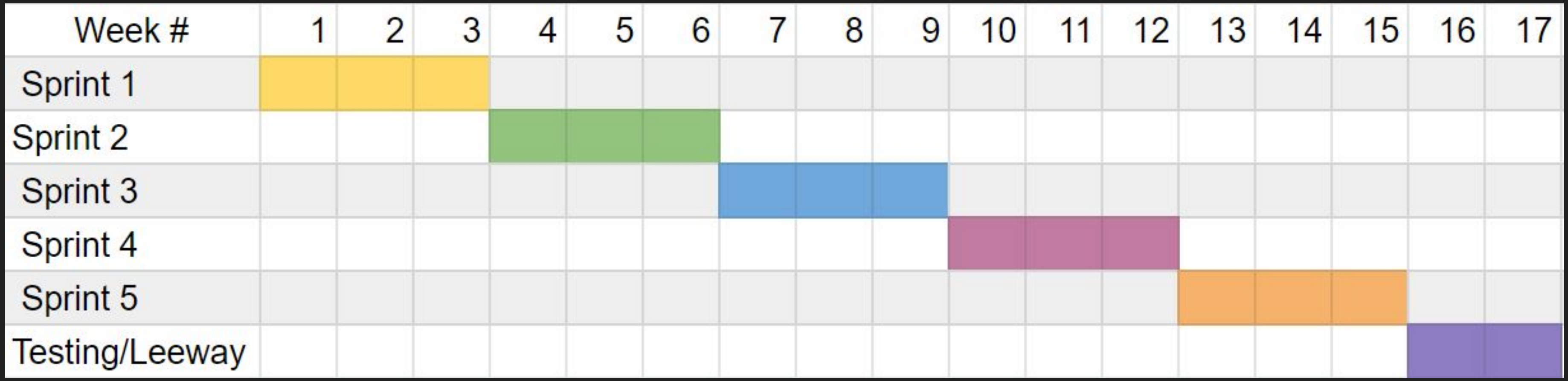


Week #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sprint 1	🟨	🟨	🟨																	
Sprint 2				🟩	🟩	🟩														
Sprint 3							🟦	🟦	🟦											
Sprint 4										🟥	🟥	🟥								
Sprint 5													🟧	🟧	🟧					
Testing/Leeway																🟪	🟪	🟪	🟪	🟪

Project Plan - Schedule/Milestones

Milestones include:

- Sprint 1/2
  - Functional graphing library
  - Asynchronous upload/read/processing of data
- Sprint 3
  - 95% reliability
  - Graphing within 2 seconds
  - Functional Backend
- Sprint 4
  - 99% reliability in Front/Backend
- Sprint 5
  - 90% test coverage



Week #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sprint 1	🟨	🟨	🟨														
Sprint 2				🟩	🟩	🟩											
Sprint 3							🟦	🟦	🟦								
Sprint 4										🟥	🟥	🟥					
Sprint 5													🟧	🟧	🟧		
Testing/Leeway																🟪	🟪

Test Plan (1-3 slides)

- How is testing performed? Simulation, emulation, actual platform, hybrid?
- Component/Unit testing (tied to requirements)
- Interface/integration testing
- System level testing/Acceptance testing – justified w.r.t. requirements
- **1-3 slides, 2-3 minutes**

Test Plan

- Unit Testing
  - React UI: Filters produce appropriate queries.
  - VMD: Presents and displays the correct data.
  - DB: Handles large amounts of users and data.
  - Algorithms: Produce the necessary calculations.
- Interface Testing
  - Frontend can complete test scenarios and produce expected results.
  - UI and VMD can send and receive data from emulated backend.
- Integration Testing
  - Data packaged by frontend can be unpackaged by backend.
  - Algorithmically outputted data can be inputted into VMD.
- System Testing
  - Test scenarios can be completed by full system to expected results.
  - Optimization of complete system to meet size and speed requirements.
- Regression Testing
  - System doesn’t lose functionality.
  - System is maintained throughout iterations.

Testing Software

JEST & React Testing Frameworks

Frontend Testing: React UI and VMD

- Unit Testing
  - Test cases for filtering and querying
  - Test cases for data displaying
- Interface Testing
  - Emulating Backend
  - Test scenarios
- Integration, System, & Regression Testing
  - Debugging
  - Completeness testing

.Net & C# Testing Libraries

Backend Testing: Algorithms and Database

- Unit Testing
  - Test cases for reading and writing data
  - Test cases for algorithm correctness
- Interface Testing
  - Emulating frontend
  - Test scenarios
- Integration, System, & Regression Testing
  - Debugging
  - Completeness testing

Acceptance Testing

Acceptance Testing -

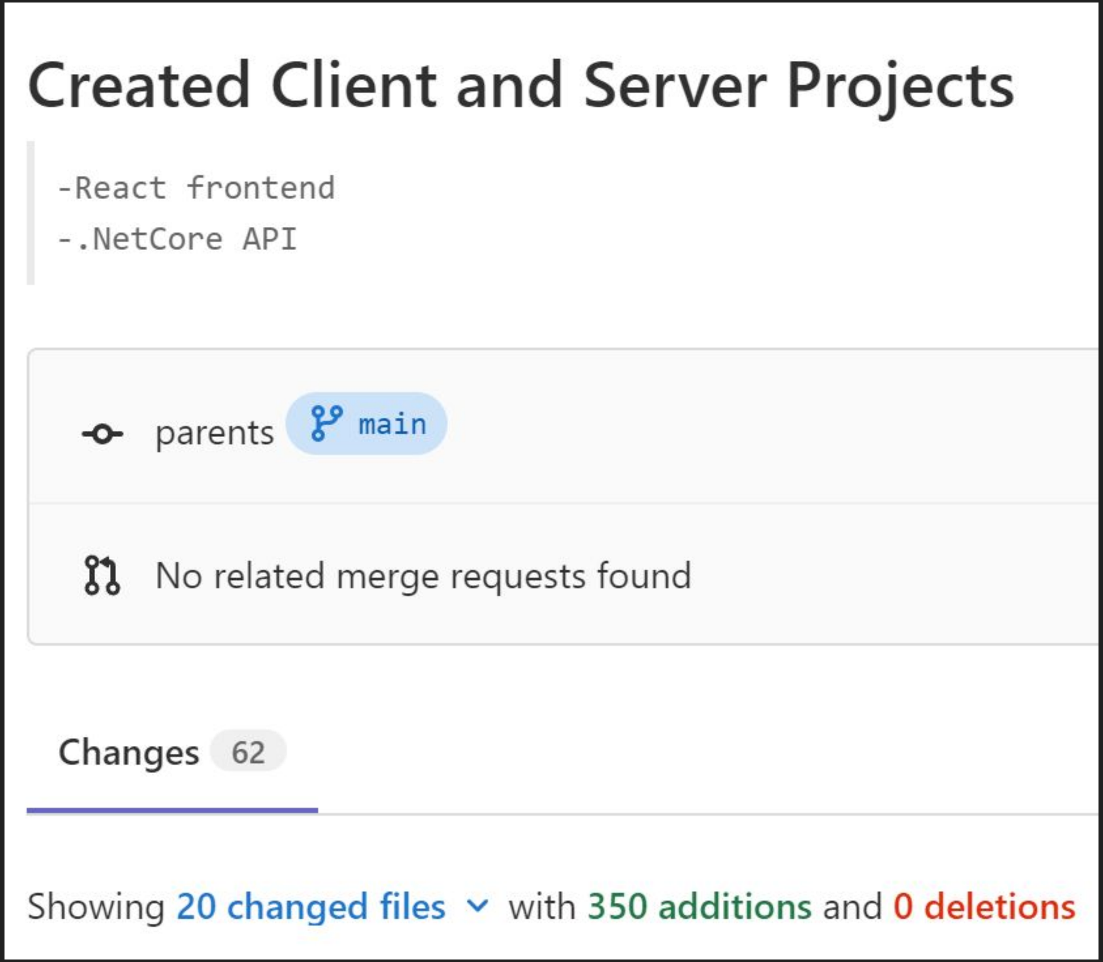
Testing the final iteration of the project alongside the clients to ensure that they can input queries and receive data that’s useful to them.

Sensitive to client requirements: UI is intuitive, System can handle large amounts of users and data reasonably quickly, filtering options are useful and relevant, visuals are easy to manipulate and understand, no bugs or errors to be found.

Can be software assisted, adjustments can and likely will be made.

Project has been completed to satisfaction.

Conclusion



Created Client and Server Projects

```
-React frontend
-.NetCore API
```

parents main

No related merge requests found

Changes 62

Showing 20 changed files with 350 additions and 0 deletions

Team Roles (Spring 2023)

Claira- Testing
Daryl- Backend
Jason- Frontend
Jeremy- Frontend

Conclusions

- Where are you in your schedule?
- Next semester’s plans
- Individual team members contributions delineated
- **1 slide, 1-2 minutes**

Questions?