Detecting and Predicting Clusters of Evolving Binary Stars

Adam Corpstein, Becker Mathie, Ethan Vander Wiel, Joel Holm, Philip Payne, Willis Knox

Adviser/Client: Goce Trajcevski

Agenda

- Team Introduction
- Project Vision
- Requirements and Constraints
- Project Plan
- System Diagrams
- Schedule and Milestones
- Testing
- Conclusion

Team Introduction

Team Roles

Becker Mathie - Chief Engineer

Joel Holm - Facilitator

Ethan Vander Wiel - Test Engineer

Philip Payne - Quality Assurance

Willis Knox - Scribe

Adam Corpstein - Report Manager

Project Vision

Problem Statement

Problem

- Organize binary-stars into groups (clusters)
- Predict future states of data

Challenges

- Multiple clustering algorithms
- No-other-way but simulation
- Big datasets

Solution

- App to track stellar evolution
- Cluster binary star systems
- Aimed at astrophysicists
- Cluster formation detection over time
- Predict stellar co-evolution

Conceptual Diagram

- User enters from our site url (no login)
- Front end will have 4 main aspects
  - Variables
  - Formulae
  - Weights
  - Output*
- Inputs our processed by our backend through a rest-api

Requirements and Constraints

Functional Requirements

- Selection of attributes (i.e. Luminosity, Mass)
- Select cluster algorithm (i.e. DBScan, K-means)
- Select preprocessing (i.e. Noise reduction, scaling)
- Simulation of future star properties
- Select time range
- Visualize data

Non-Functional Requirements

- Efficient cluster request responses (< 20 seconds)
- Simultaneous user count (< 1000)
- Scalable databases
- Environmental requirement: Internet access
- Economic requirement: Personal computer

Constraints and Assumptions

- Required background knowledge
- Data complexity
- Capabilities of clustering algorithms
- Features (data types) present in database

Project Plan

Risks and Mitigation Plan

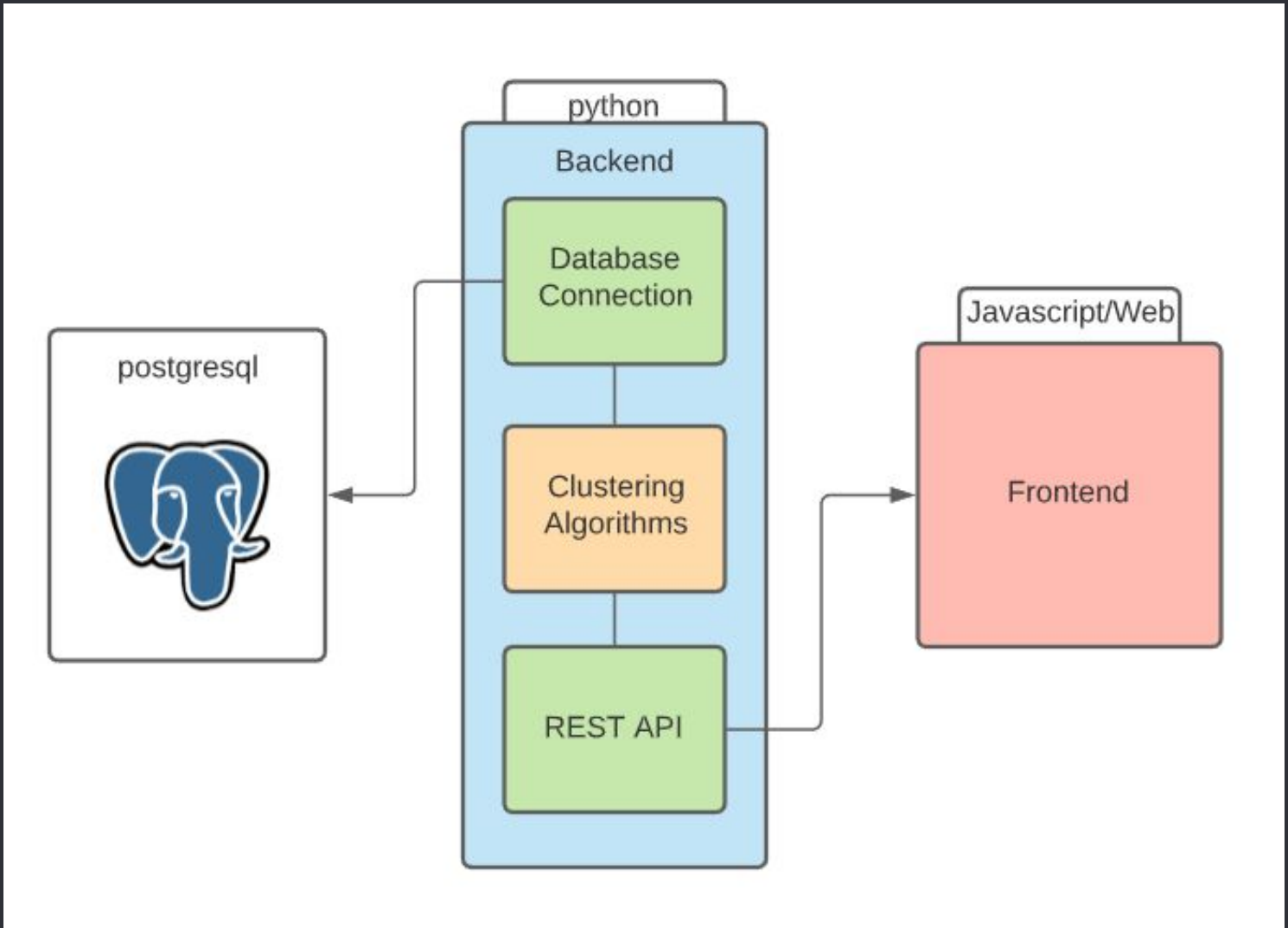
- Additional clustering algorithms
  - Focus on extendable architecture
- Alternative data visualization
  - Versatile Javascript graphics libraries
- COVID-19
  - Social distancing practices
  - Focus on communication

Technology Stack

- Django Web Framework
  - COSMIC
  - scikit-learn
  - PostgreSQL
- Angular Framework/Angular Material

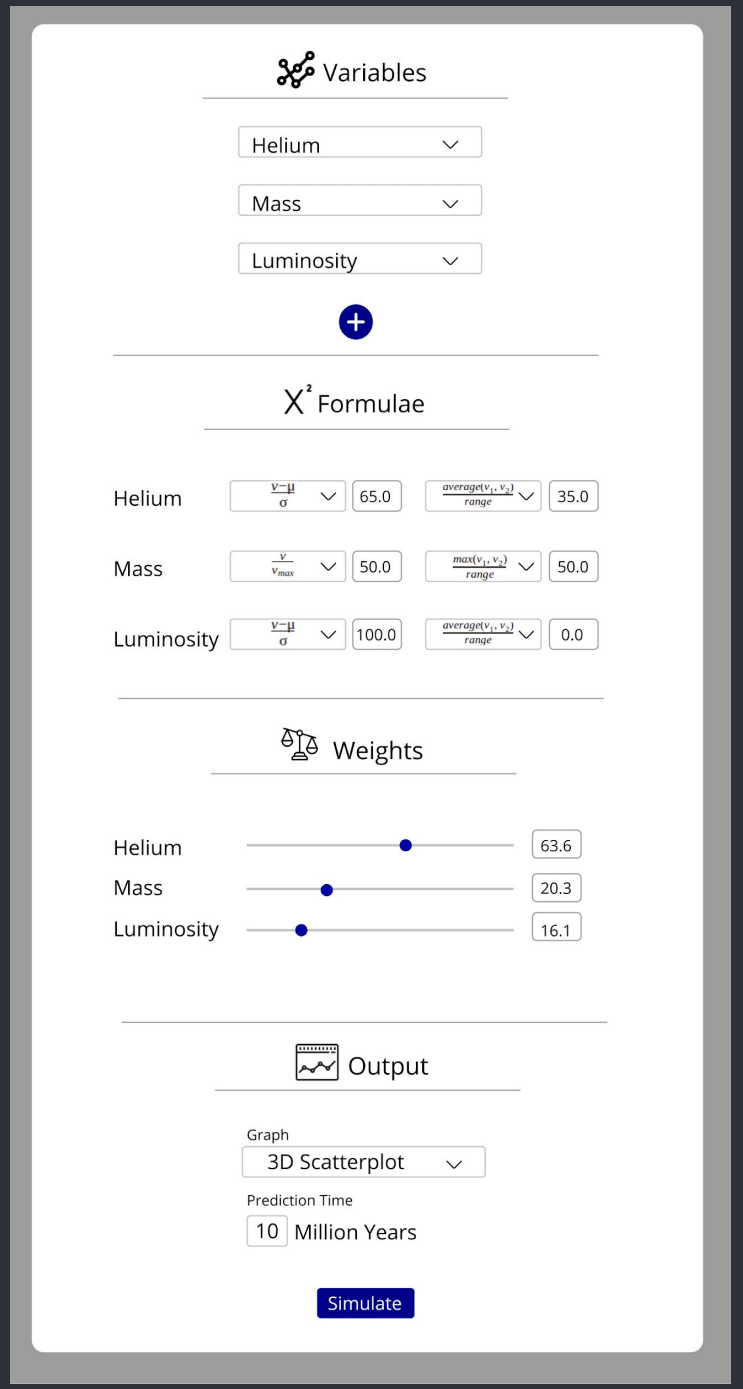
System Diagrams

High Level Architecture

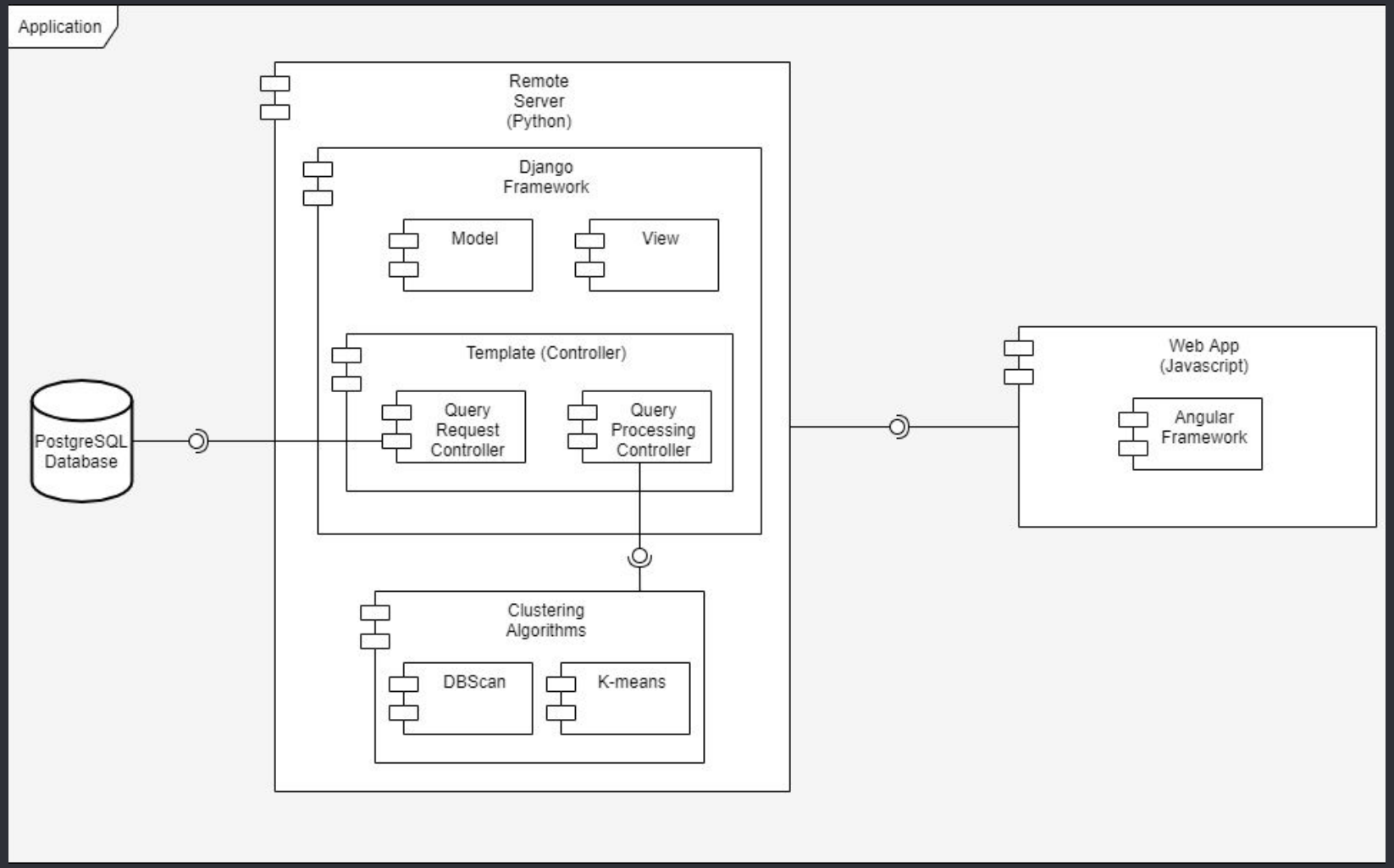


Preliminary UI

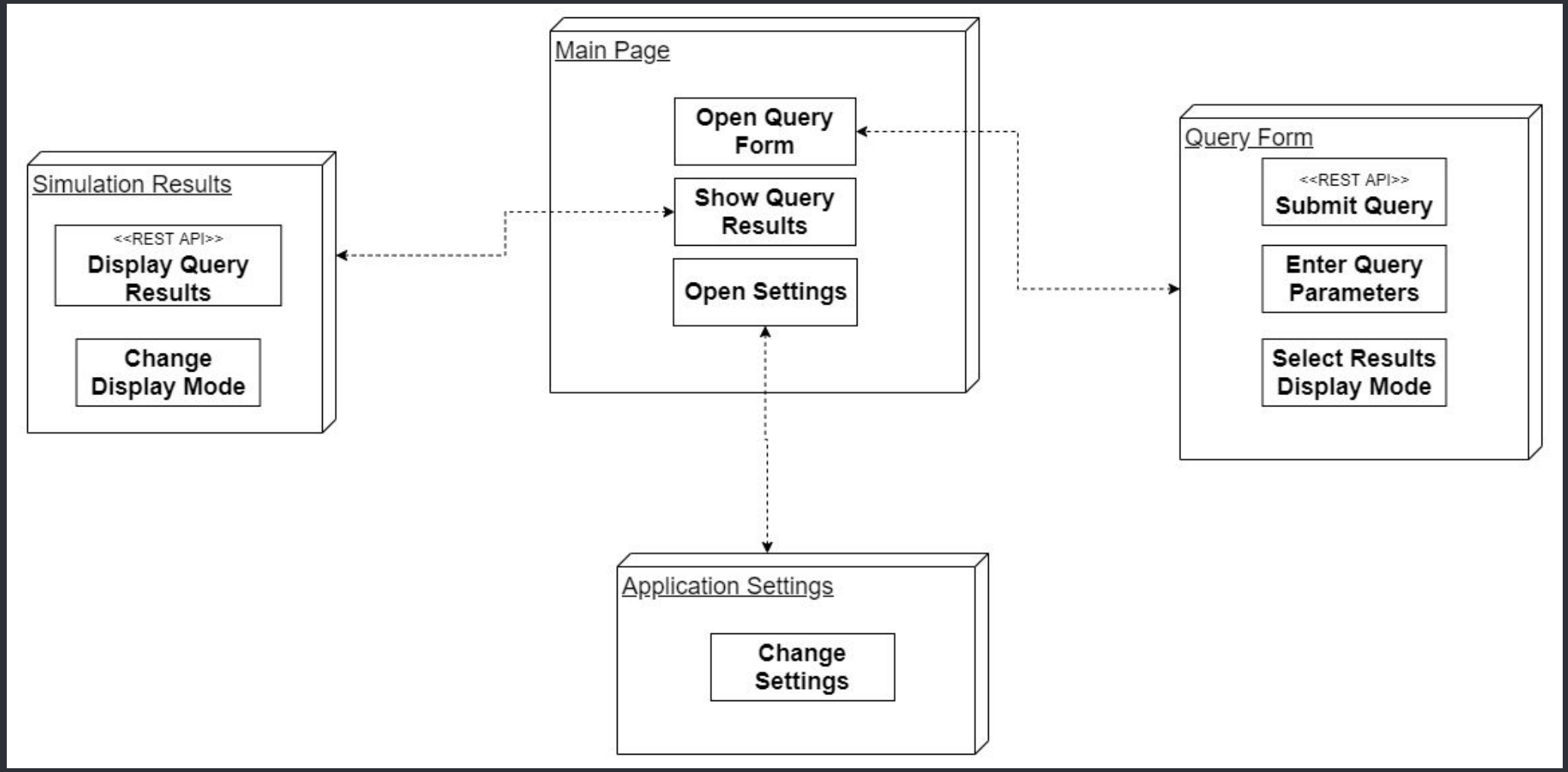
- User selects attributes
- Formulae & weights rely on variables
- Adaptive UI



Component Diagram



Frontend Design



Schedule and Milestones

Progression Metrics

- Algorithm Choice
- Determine Software
- Choose Distance Functions
- Preliminary Query UI
- Round Trip Communication
- CI/CD
- Clustering by one data type
- Graphical Display
- Useable alpha/beta stages
- Cluster by multiple data types
- Clustering Evolution

Amplitude Scaling	Offset Translation	Divide by Maximum Value	Minmax Normalization	Ratio Between Stars	Average Over Range	Difference Over Range
$\frac{v-\mu}{\sigma}$	$v-\mu$	$\frac{v}{v_{max}}$	$\frac{v-min}{max-min}$	$\frac{v_1}{v_2}$	$\frac{average(v_1, v_2)}{range}$	$\left\lvert\frac{v_1-v_2}{range}\right\rvert$
Normalizing	Normalizing	Normalizing	Normalizing	Normalizing	Non-normalizing	Non-normalizing

Milestone Schedule

Planning and Preparation
Aug 31 - Oct 25

Research

Use Cases

Algorithm Choice

Development
Oct 26 - Apr 5

Front End UI

Mockups

Alpha Release

Beta Release

Modifications

Testing

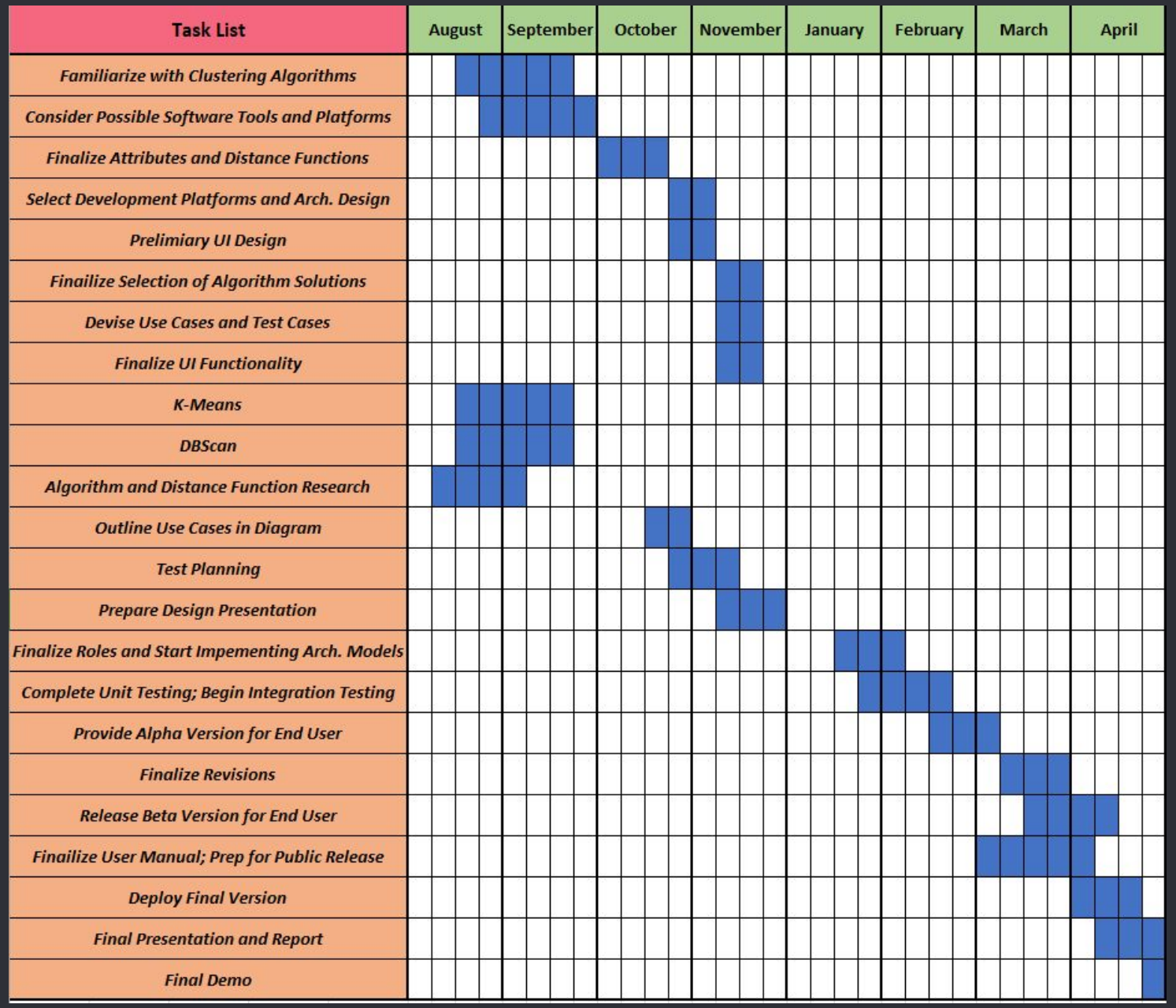
Final Release
Apr 6 - Apr 30

Collect Feedback

User Manual

Product Release

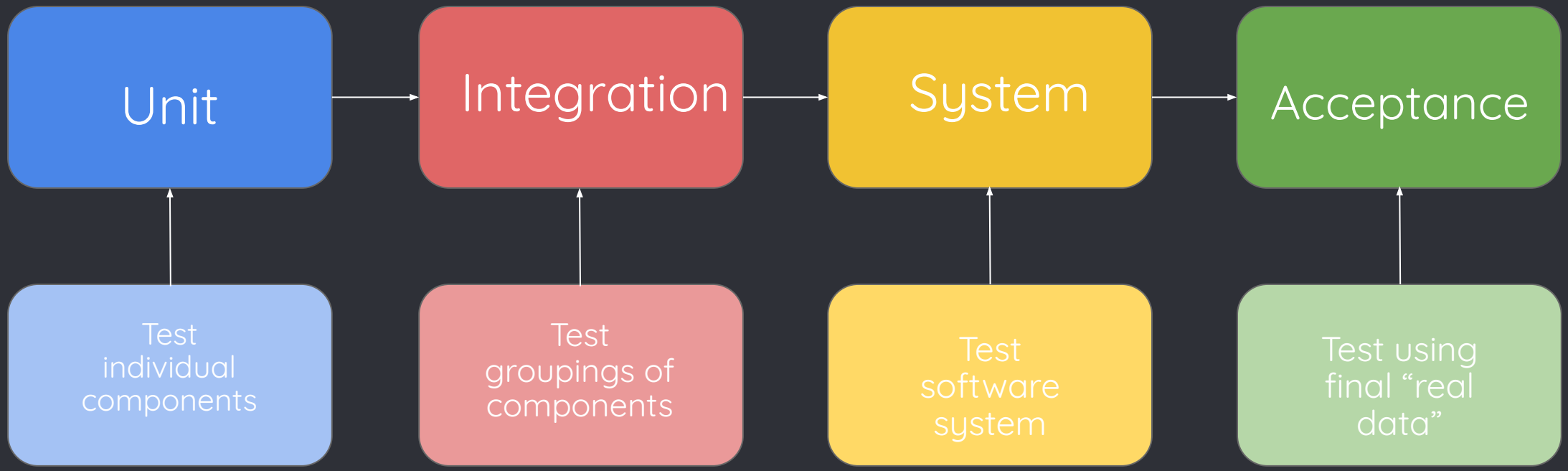
Full Schedule



Testing

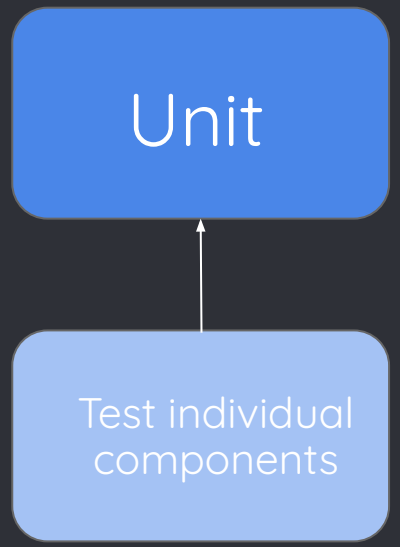
Test Plan - Overview

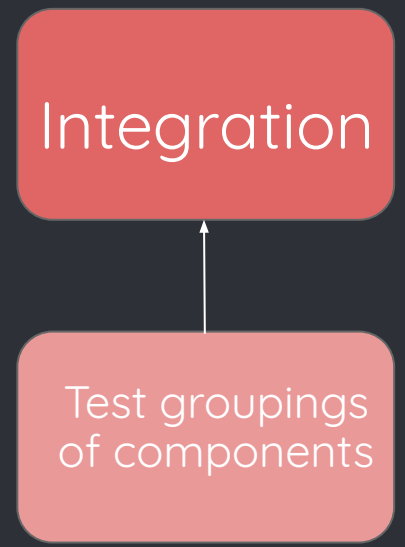
- Modern software testing approach



Unit and Integration Testing

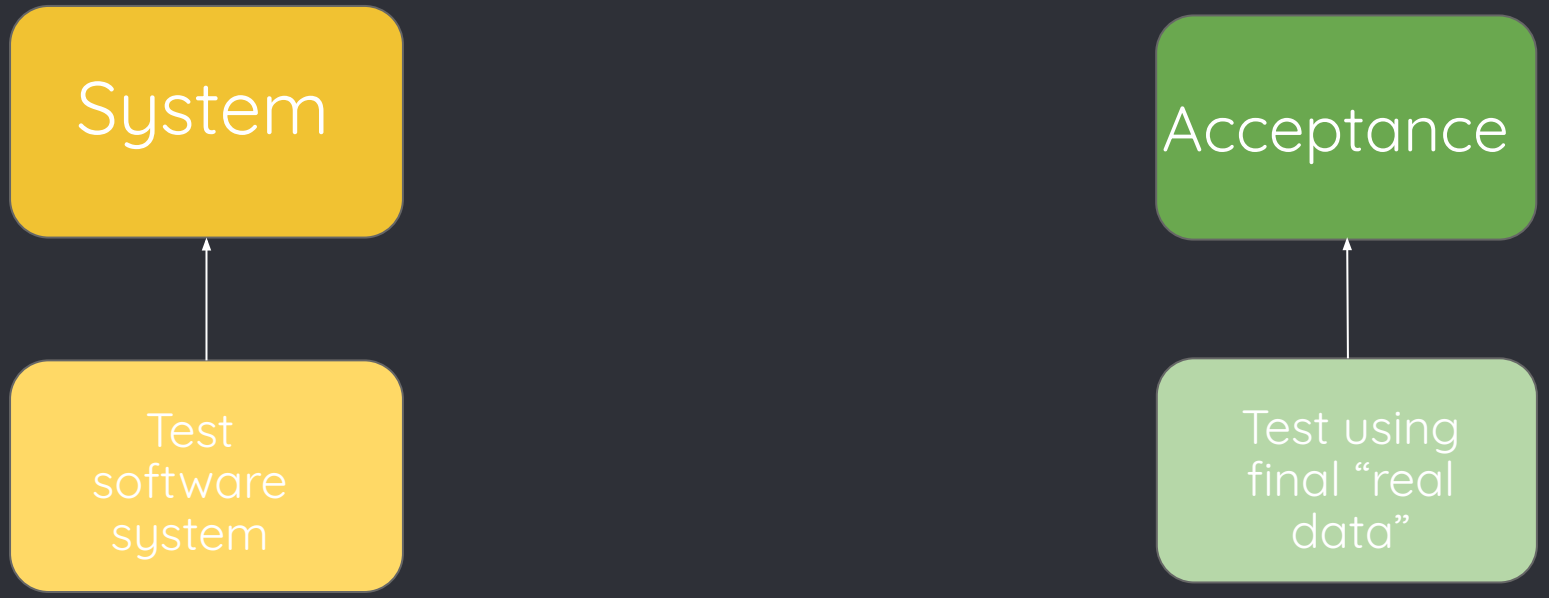
- Tests using “mock” data
- Test UI and Backend (separately)
- Tests our functional requirements





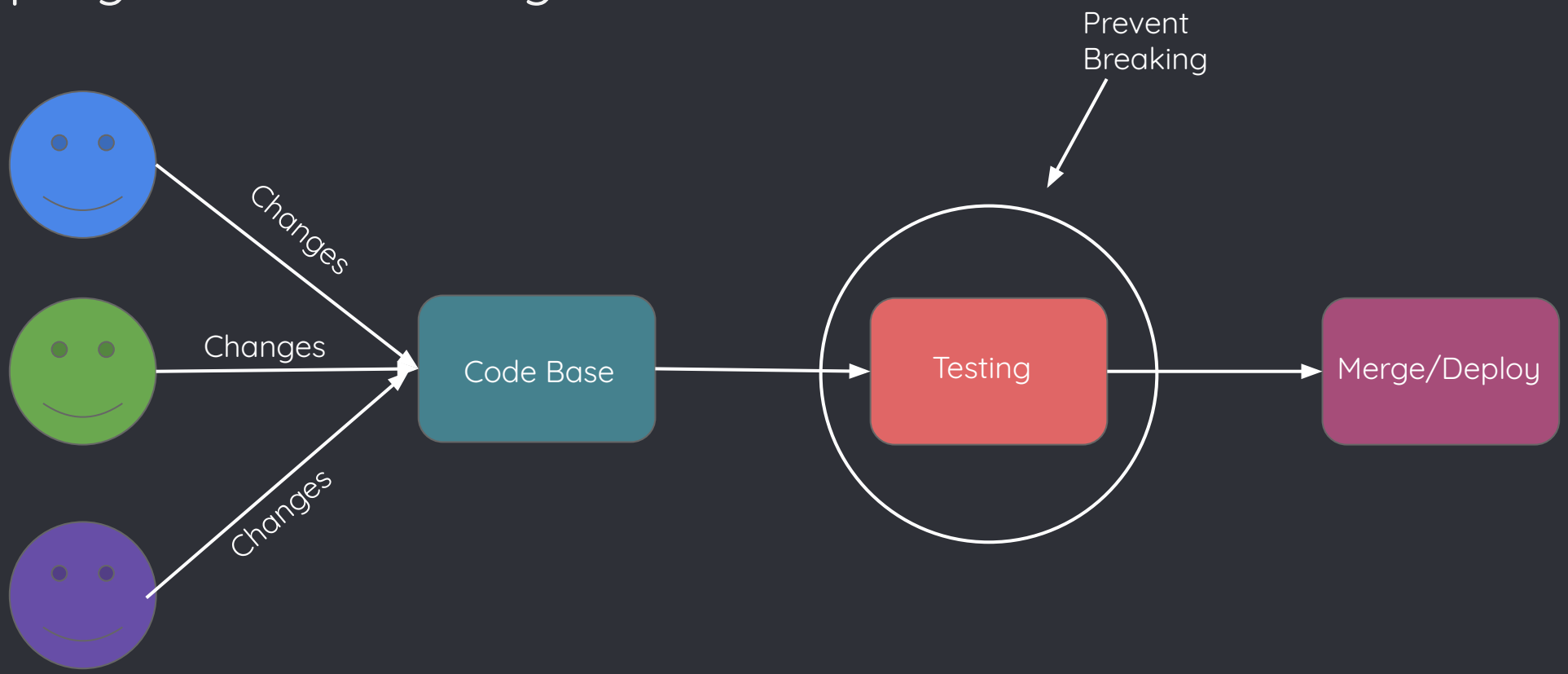
System and Acceptance Testing

- Tests entire system
- Communication between subsystems
- Finalizes customer requirements
- Automatically maintain changes to production



CI/CD

- Ensures new code compliance
- Deploys automatically



Conclusion

Next Steps

- ~~Research, design, technology choices, DB~~
- Reserve server space
- Start UI and backend implementation in Django
- Set up automated tests and CI/CD
- Move backend to new server space

Individual Contributions

- **Adam Corpstein** - Team Website, Documentation/Azure Website
- **Becker Mathie -** Project Timeline/planning, Determining technologies, Test planning
- **Ethan Vander Wiel** - UI Design, Distance Functions, Testing lead
- **Joel Holm** - Functional/Non-Functional Requirements, Constraints and Assumptions, Risks & Mitigation
- **Philip Payne -** Architecture design, UML diagram design, Quality control
- **Willis Knox** - DB implementation, Distance Functions, Meeting Notes

Each member helped with general assignments, presentations, and report writing

Q&A