MICHIGAN STATE UNIVERSITY

Project Plan

Review Aggregator for Educational Programs

The Capstone Experience

Team Malleable Minds

Cody Carter
Jerry Cortez
Becky Henning
Brian Martin

Department of Computer Science and Engineering
Michigan State University

Fall 2020

From Students…
…to Professionals

Functional Specifications

Customers are accustomed to using review aggregators when making buying decisions. We will utilize this familiar model to create a web app for parents making educational decisions for their children.

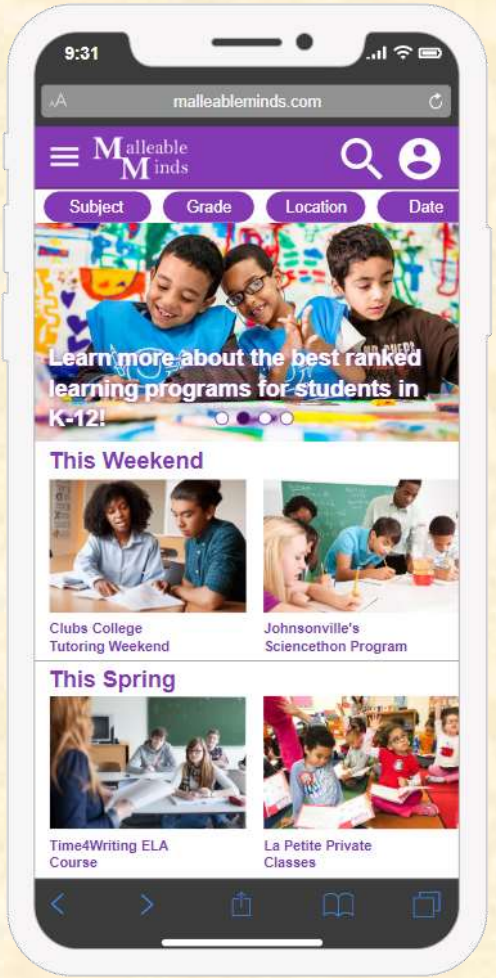
Features:

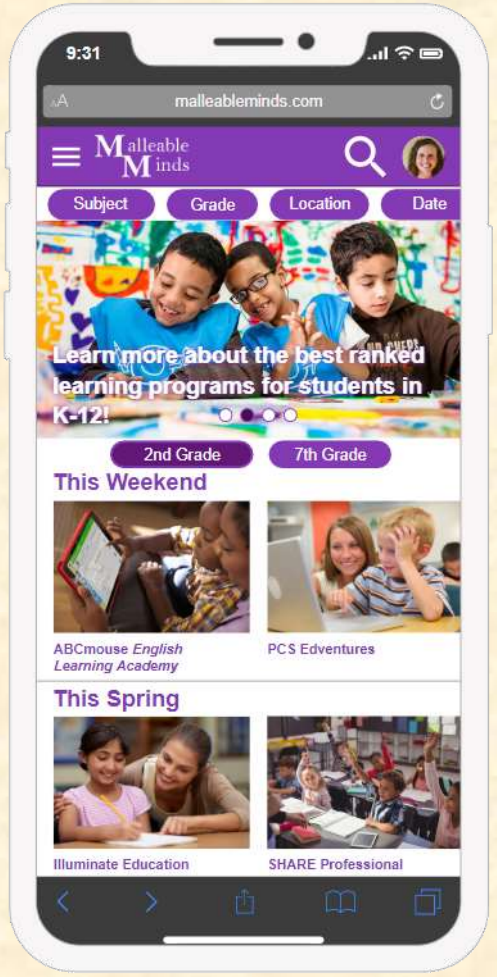
- Review aggregator
- General search capability
- User specific experience and reviews
  - Parent, Educator, and/or Provider
- Individual profile and program views
- Program likes, Educator following, personalized push notifications

Design Specifications

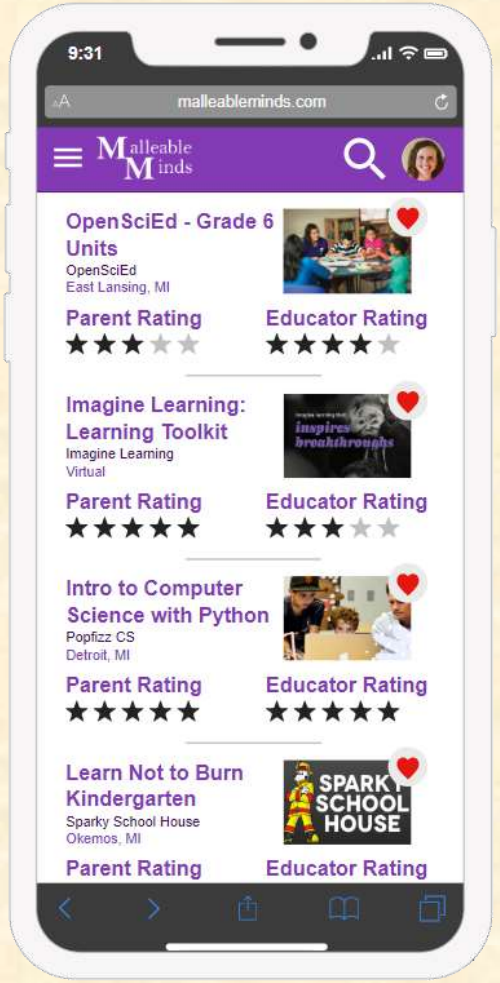
- Review aggregator web application that can be used on any web-enabled device (focus on PC and mobile internet browser compatibility)
- Search functionality to find a list of programs and look at a program's details
- Navigate different pages with a hamburger menu icon
- Different account types with account-specific features
  - Parent - tailored home page, likes, follows, notifications, suggest edits on program details, individual reviews identifying as parent
  - Educator - same as parent, but individual reviews identifying as educator
  - Provider - add/edit/remove program information, cannot review but can respond to reviews on their programs

Screen Mockup: Home Page

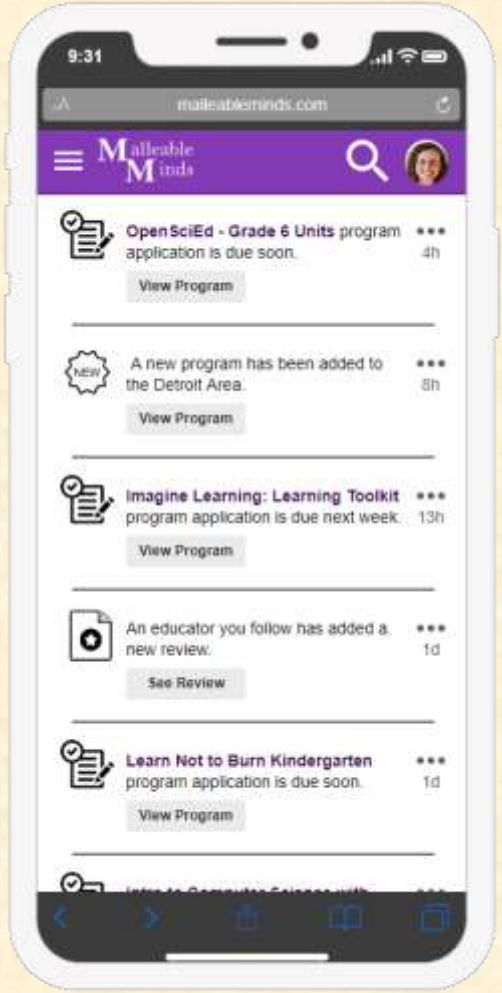




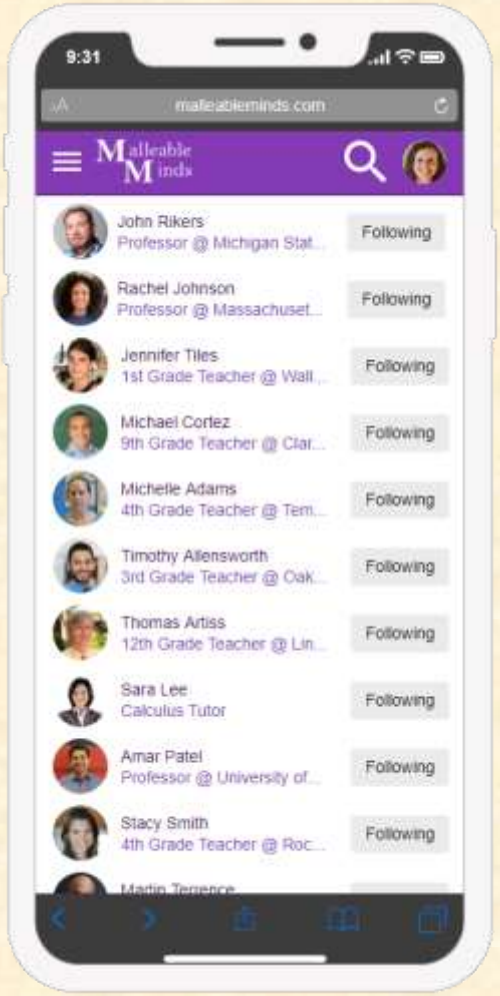
Screen Mockup: Likes Page



Screen Mockup: Notifications Page

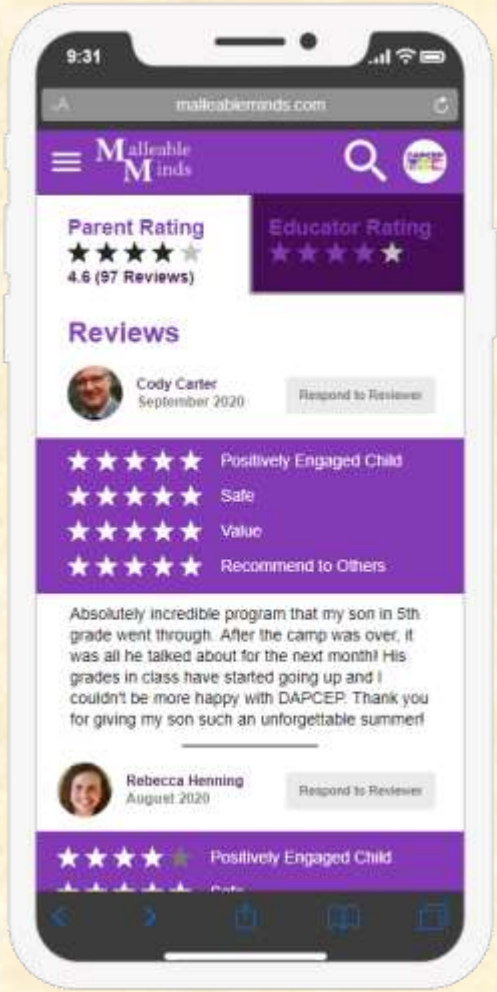


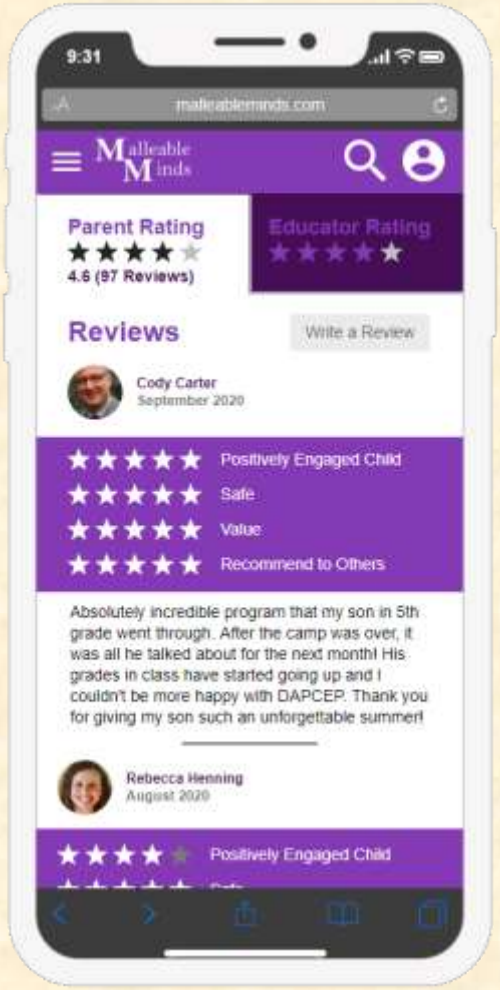
Screen Mockup: Following Page



Screen Mockup: Program Details Page

Screen Mockup: Reviews Page

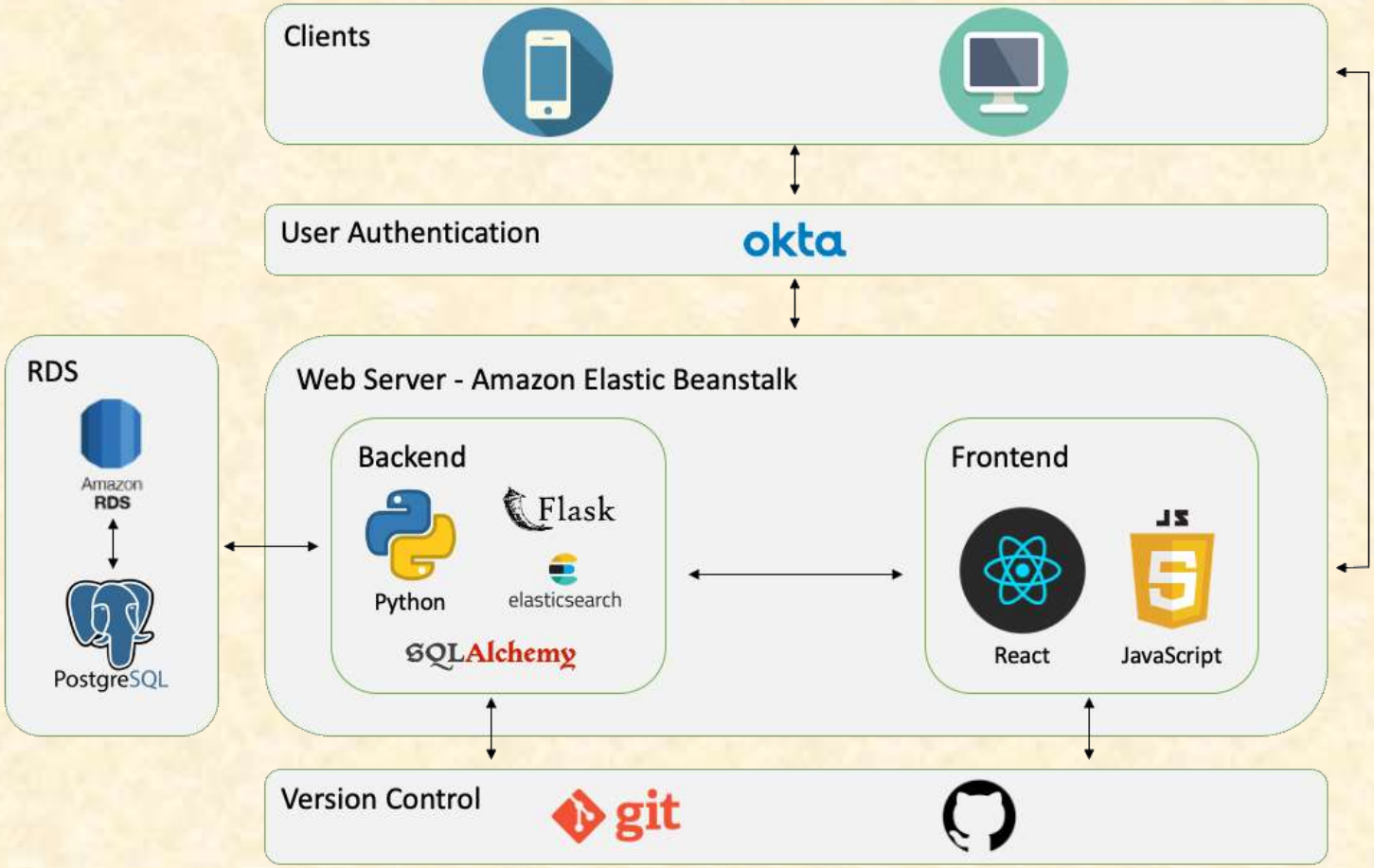




Technical Specifications

- Web-based application deployed and hosted through Amazon Web Services
  - Designed for Mobile and PC Browsers
- Leverages a Flask framework that connects the Python-based backend to the React-based frontend with the use of a RESTful API
- PostgreSQL database schema deployed through Amazon RDS
  - Accessed in backend using SQLAlchemy Library
- User Authentication verified through Okta
  - Also allows for unauthenticated “public” access
- ElasticSearch implemented in the backend for search and filtering capabilities

System Architecture



System Components

- Cloud Platforms
  - Amazon Web Services
    - Amazon Elastic Beanstalk – Web Server
    - Amazon Relational Database System
- Software Technologies
  - Frontend: React, JavaScript, Okta
  - Backend: Python, SQLAlchemy, ElasticSearch
  - Version Control: Git, SourceTree
  - Development Environment: PyCharm, Visual Studio

Risks

- Customizing Content for Multiple Children
  - Need to determine how to best present information to a parent with multiple children of varying ages
  - We will design a highly customizable experience with user filtering
- Efficient Database Storage and Query Design
  - As the app scales it will likely degrade performance, and database structural changes will be difficult once real users are involved
  - We will build in stress testing components with design as we go
- Evolving Project Requirements and Scope
  - We are building quickly and iteratively from the ground up, and when faced with unknowns we must rework and add new components
  - As we re-scope we will discuss what items must go out-of-scope in favor of reworking and replacing current features
- Initial Program Information
  - Database information will initially come from CEO, and we will not have data as we build to test our frontend/backend
  - We will work with CEO to nail down data fields well in advance and develop an automated process to upload data sets as they are delivered. We will also curate a dummy data set for testing purposes

Questions?