Team Project: Beav-Guesser

Team Info:
Provide a concise summary of the project team and project artifacts. Specifically:
List each team member and their role in the project.

- Blake - Database
- Joy - API/ data collection
- Kevin Tran - Backend
- Kevin Nguyen - Backend
- Sam - UI Designer/Dev
- Gavin - Frontend
- Lukas - API/data collection

Role Description:
- Database:
  Creation of database and integration with application
- API/data collection:
  Collection of photos and integration of Google street views
- Backend Developer:
  Creation of server and backend logic
- Frontend Developer:
  Creation of website and integration of website with backend processes and user interface
- UI designer/Dev:
  Creation of user interface and frontend structure

Communication/Tools:
Github:
https://github.com/blakethomas12/Team-18
Trello:
https://trello.com/invite/b/6785b34faf152936f7a64481/ATTI04781f6d4e3950e8c2c6ecd52edaf43a00ED59BB/pt18-beav-guesser
Discord:
https://discord.gg/cD4kHm3b

List communication channels/tools and establish the rules for communication.
Discord server
Rules:
- Communicate any changes within 6 hours of completion
- Members report their updates via Discord when they can't be present
- Give everyone a chance to talk and express their ideas
- Ask for help when there's an as soon as there is an issue with their task

Product Description

Abstract:

Many students often struggle to find campus buildings, especially at a new institution or on the first few days of a new term. The purpose of our project is to help familiarize students with campus locations in an engaging way. Our project is a take on the game Geoguesser adapted to the OSU campus. Our game will be a web-based application where users are shown a 360 image from around the Oregon State University Campus and asked to guess the location of the image taken. Our major developmental goals are a user authentication system, user profile, leaderboard system, and integrated locational data.

Goal:

The goal of this project is to help students engage more with the OSU campus by playing a game while familiarizing themselves with the different classrooms and buildings. With our leaderboard and scoring, students will want to compete against others to improve their campus knowledge. Having an interactive game that has every location of the campus and constantly showing the same buildings will help students recognize different locations, which makes navigating through campus easier.

Current practice:

Currently, students are getting familiarized with the campus through many ways. One way is through the tours that we often see around campus. Another way is by simply asking people around campus, where certain locations are. However, the most common way is using GPS. Students often use either Apple Maps or Google Maps, which helps provide live directions to the buildings on campus.

Novelty:

This would be a project that works on a more local scale, one that is more applicable to the current living environment that the group and other students at OSU are currently familiar with, thus encouraging a more observant approach to everyday student life. Our project offers a fun and engaging way to learn more about the campus that the current practices don't have. Being able to familiarize themselves through a game helps keep the students inclined to want to come back and play, thus improving their campus knowledge.

Effects:

Anyone at Oregon State University will seek this game for simple, but engaging entertainment. If the product is successful, students will have more knowledge about the OSU campus and its landmarks. Currently, this project is specifically for Oregon State, hence the Oregon State theme, but in the future we could possibly expand to other campuses with a few changes to our code.

Technical approach:

To use Beav-Guesser, users can navigate to our website at https://beavguesser-095f2d551c81.herokuapp.com/ . Then, they should see our home page and with a “Play Game” button and a navigation bar at the top with a “Home”, “About”, “Leaderboard”, “Login”, and “Signup” buttons. The user should sign up or log in to track their scores and show up on the leaderboard. Once the game begins, users are shown a 360 map of a random place on campus. To make a guess, the user will click on the Show Map at the bottom left corner and make a guess by pinpointing a spot on the map. A “Next Round” button will pop up to move on to the next round. After doing this for 5 rounds, the user is prompted with a final score and a “Play Again” and “Home” button.

For all of this to work, we will create a website connected to a custom database of locations and the Google Street View API. We serve a random picture or location from the database to the user. This is linked to a point on the map. The user inputs a point on the map. This is analyzed, and the user is given a certain amount of points for how close they are. These points are stored in the database and added to the leaderboard, which is displayed at the end of the game.

Risks Assessment:

1) Risk: There may be limits to the Google Maps API
Risk Level: Low
Impact: Medium
Evidence: Too many requests during testing
Steps: Make meaningful contributions during testing and be aware of the amount of requests made
Detection Plan: We will monitor API requests and log it to prevent going over the limit
Mitigation Plan: Halt testing that involves the API and continue implementing features since locations are stored in a database

2) Risk: Hosting and Deployment Challenges
Risk Level: Medium
Impact: High
Evidence: Server configuration issues, or cloud provider limitations could lead to downtime or failed deployments. Unexpected costs or resource constraints may also impact performance.
Steps: Only deploy working versions of the application onto the cloud.
Detection Plan: We will rerun the website multiple times and on different devices to check if our application still works on the cloud.
Mitigation plan: Most testing and implementations will be done on the localhost server. Any working version that will be shown to our customer will be deployed on the cloud beforehand.

3) Risk: Data Retrieval Issues
Risk Level: Medium
Impact: High
Evidence: potential API limits and implementation of incorrect data retrieval methods.
Steps: Test data retrieval from the database and revise if incorrect .
Detection Plan: Frequently test data retrieval and implement data validation checks.

Mitigation plan: optimize database queries

4) Risk: Frontend-Backend Communication Issues
   Risk Level: Medium
   Impact: High
   Evidence: Incorrect API responses or misaligned request formats may cause the game to break
   Steps: Implement logging for API request and reasons, and conduct frequent integration tests.
   Detection Plan: Implement logging for API request and reasons, and conduct frequent integration tests.
   Mitigation plan: Display error messages when the back-end fails.

5) Risk: Map Loading Failures
   Risk Level: Medium
   Impact: High
   Evidence: User may not be able to play the game if the map fails to load due to javascript.
   Steps: Ensure error handling works in JavaScript
   Detection Plan: We will monitor console error messages.
   Mitigation Plan: Add reload map function.

Project Schedule:

Identify milestones (external and internal), define tasks along with effort estimates (at granularity no coarser than 1-person-week units), and identify dependences among them. (What has to be complete before you can begin implementing component X? What has to be complete before you can start testing component X? What has to be complete before you can run an entire (small) use case?) This should reflect your actual plan of work, possibly including items your team has already completed.

To build a schedule, start with your major milestones (tend to be noun-like) and fill in the tasks (tend to start with a verb) that will allow you to achieve them. A simple table is sufficient for this size of a project.

Milestone	**Tasks**	**Effort Estimate**	**Dependency**
Setup Project	Establish team rules, assign team roles, set up everyone's systems, setup GitHub	1 week	None
Frontend	Develop UI design,	3 weeks	Project planning

Development	Interactivity and client-side logic		complete
Backend Development	Setup server side, connect to database, set user authentication, game logic	4 weeks	Frontend and database development must be started
Database Development	Create tables for user, leaderboards, and guesses	2 weeks	Project planning complete
Debugging/Testing/ Changes	Do a complete run and walkthrough on the website, fix any changes needed	2 weeks	All of development must be complete.
Demonstration	Prepare documentation and presentation	1 week	Project is complete and ready.

Test plan & bugs:

Describe what aspects of your system you plan to test and why they are sufficient, as well as how specifically you plan to test those aspects in a disciplined way. Describe a strategy for each of unit testing, system (integration) testing, and usability testing, along with any specific test suites identified to capture the requirements.

We require that you use GitHub IssuesLinks to an external site. to track bugs that occur during use and testing.

Aspects we plan to test: Account Creation, Account Management, Starting a new round with valid location, Submitting a guess, Leaderboard Functionality, UI on Home and About page.

The testing will be done via testing suite files, which when run, test individual functions as well as full functionality of the program. These test suites will be divided into four files. One file will be used for full system testing and the other three will be used for user to website, website to server, and server to database respectively. Usability testing will be done by asking friends to use and navigate the program without prior information, and by having other members of the group attempt to use the feature given that they did not work on the feature themselves. System testing will be done through one main file, while Unit testing will be done within the other three test suites to test individual functionality.

Currently, we plan to write system tests for most of the account creation functionalities like registration and login buttons. We also will create system tests for updating your username and password. We will also have system tests for the map correctly loading, the user guess submission, and the leaderboard updating the top scores.

The test-automation we are using is Jest. We choose this as all of our logic is written in JavaScript and Jest is for creating tests for JS and is easily used and installed with npm. Adding a new test is easy. You can add a new file to the __test__ folder which is run by Jest. You can also add a test to one of the existing files. Additionally, we are using Github Actions as our CI service. When the CI is run, a security analysis is performed on the code. On top of that, all the unit tests in Jest will also be run. Any time a member of the team pushes to the main branch on Github it triggers the CI to run.

	Github Actions	Jenkins CI	Travis CI
Pros	Works directly in repo and Github, highly customizable, real-time feedback	Open source, platform independent,	Easy to use, automated testing, free for open source
Cons	Complexity, resources limits, dependency on Github	Complex setup , regular maintenance, resource intensive	Performance issues, limited customization

Documentation Plan:

Outline a plan for developing documentation that you plan to deliver with the system, e.g., user guides, admin guides, developer guides, man pages, help menus, wikis, etc finish the documentation plan.

To allow for ease-of-use and for new developers to come into this project we will be providing these documents.

1. User Guides
    a. Audience: Users/Customers
    b. Content: There will be a small overview of the game at the beginning before you click play. This overview will include how to play the game and how points will be accumulated. For more information on the leaderboard and account creation, there will be a help page that you can click.
2. Administrator Guide
    a. Audience: Admins and Maintenance Users
    b. Content: There will be a pdf within our GitHub repository that outlines how to manage the User profiles, like the passwords. It will show ways to update the leaderboard and to add/update locations within the application. There will also be information on how to back up the database.
3. Developer Guide
    a. Audience: Developers

b. Content: This section will be a guide in the form of a pdf within our GitHub repository. This guide will show how to set up the Beav-Guesser and show the changes within the web server. This includes how to run it locally or on the web server. It will also show the code structure and where every process is done within the code. Our API will be documented with its end dates of use, and our database schema and structure will be included. Rules will be in place if there are any changes or a new developer is brought in.

Use Cases:

1. Title: Changing your username
   Written by: Blake
   a. Actors: User
   b. Triggers: Change of Username
   c. Preconditions: User exists, new name is different than old name, new name is not taken
   d. Postconditions (success scenario): Username has been changed
   e. List of steps (success scenario)
      1. User requests change of username
      2. User asked for new username
      3. Check if old username = new username, reprompt if true
      4. Check if new username is taken, reprompt if true
      5. Username passes checks and is updated in database
   f. Extensions/variations of the success scenario
      1. User fails check if old name = new name, then corrects name
      2. User fails check if name is taken, then enters a free name
   g. Exceptions: failure conditions and scenarios
      1. User does not exist
      2. Database is unreachable(on update or checks)
2. Title: Moving onto a new round
   Written by: Lukas
   a. Actors: User
   b. Triggers: Start new round
   c. Preconditions:
   d. Postconditions (success scenario): User is served location
   e. List of steps (success scenario)
      i. User starts new round
      ii. Random location is pulled from database
      iii. Corresponding streetview is requested from API
      iv. Correct streetview is shown to user
   f. Exceptions: failure conditions and scenarios
      i. Database is unreachable
      ii. API is unreachable

        iii. Incorrect location
        iv. Incorrect streetview

3. Title: Submitting your guess
   Written by: Kevin Tran
    a. Actors: User
    b. Triggers: Submits Location
    c. Preconditions: The round has started and user is presented with the location
    d. Postconditions (success scenario): User guess is recorded and tallied
    e. List of steps (success scenario)
        i. User clicks a location on the map and submits a guess
        ii. Backend received the guess coordinates and the correct coordinates
        iii. Backend calculates the distance between two locations
        iv. Points are rewarded and the user’s score is stored in a database.
    f. Extensions/Variations of the success scenario
        i. User are shown two points on the map that correspond to their guess and the correct answer.
    g. Exceptions: Failure Conditions and Scenarios
        i. Invalid Guess
        ii. API unreachable
        iii. Database Unreachable
4. Title: Viewing the leaderboard
   Written by: Joy Lim
    a. Actors: User
    b. Triggers: View Leaderboard
    c. Preconditions:
        i. System has recorded scores for at least one player
        ii. Leaderboard data is up-to-date
    d. Postconditions (success scenario):
        i. Leaderboard option is displayed on the main page
        ii. Leaderboard will display the rank, username, and score
    e. List of steps (success scenario)
        i. User selects “Leaderboard” option from the main page
        ii. System will retrieve user scores from leaderboard data
        iii. Scores will be sorted in descending order
    f. Extensions/variations of the success scenario
        i. User leaves before scores are recorded
    g. Exceptions: failure conditions and scenarios
        i. Leaderboard is empty meaning the database contains no scores
5. Title: Logging in to game
   Written by: Gavin Fifer
    a. Actors: User
    b. Triggers: Login
    c. Preconditions: no account is currently logged in
    d. Postconditions (success scenario): The user is logged into their account

e. List of steps (success scenario)
   i. User clicks the Login button
   ii. System will display the Username and Password fields
   iii. User fills in Username and Password fields and clicks Submit button
   iv. System validates the Username and Password entered
   v. System returns User to the main page now logged in

f. Extensions/variations of the success scenario
   i. The Login button is now replaced with the Logout button
   ii. The User can see their Username next to the Logout button

g. Exceptions:
   i. Invalid Username/Password
   ii. The account is already logged into somewhere else
   iii. Database with Usernames/Passwords cannot be reached

6. Title: Creating a new account
   Written by: Sam
   a. Actors: User
   b. Triggers: New user registration
   c. Preconditions:
      i. User isn't currently logged in
      ii. User provides valid registration details
   d. Postconditions (success scenario):
      i. User account is created and stored in the database
   e. List of steps (success scenario):
      i. User selects the "new account" button from the menu
      ii. System prompts user for username, email, and password
      iii. User enters details and submits
      iv. System checks if submitted details already exist in database
      v. The check passes and the new account is created and stored
      vi. The user is automatically logged into their account
   f. Extensions/variations of the success scenario:
      i. User inputs an option profile picture
      ii. A "forgot password" routine is initiated
   g. Exceptions:
      i. Username/email already exists in database
      ii. Database is currently unavailable

7. Title: Changing your password
   Written by: Kevin Nguyen
   a. Actors: User
   b. Triggers: Updates password
   c. Preconditions:
      i. User account already exists
      ii. User provides the current password for the account
   d. Postconditions (success scenario): User is able to update their password
   e. List of steps (success scenario)

        i. User clicks on “Update Profile” in the profile page
        ii. System asks for current password
        iii. User enters in the current password
        iv. User enters in a new password and clicks on “Confirm”
        v. System verifies that current username and password exists in the database and updates the new password to the database
        vi. User is back in the profile page and clicks on “Log Out”
        vii. User enters username and new password and clicks “Log In”
    f. Extensions/Variations of the success scenario
        i. User enters an invalid current password and a message pops up saying to try again
    g. Exceptions: Failure Conditions and Scenarios
        i. Database is down
        ii. Current password is not found

Non-Functional Requirements:

1. Databases must be able to grow and serve a growing user base.
2. Photos must not include people's identities
3. Front end must be easy to use and simple to learn

External requirements:

- Since our product is web based the url must be public and accessible to others
- Our project will also include a set of instructions to set up a server and start to program

3. Additionally, add the following:

Major Features:

- Location Data
- Leaderboard
- User Authentication
- User Profiles

Stretch Goals

- Multi-User competition
- User Submissions
- Reward System

4. Timeline

Week 1: project idea, role creations, research into roles, tools and common practices
Week 2: basic foundation of website created(basic server uses, interactable front end)
Week 3: database and api integration, data gathering, game logic
Week 4: major features implementation, data gathering
Week 5: finishing major features, data gathering
Week 6: Final touches to all parts, adding in the smaller details(ui fixes, visual components, cleaning up code)
Week 7: Major testing, test all components for bugs
Week 8: Major testing, test all components for bugs

Week 9: Final testing, preparation for final version

5. Software Architecture
The three main components for our project are a server, database, and frontend.

Server:
The server is the main place of communication between all the rest of the parts. The client first requests files from the server. Then game data is passed back and forth through the server to calculate scores, store final scores, and get photospheres.
Roles:

- Folder with frontend files to serve
- Routes client requests for backend functions
- Process API request for user authentication and game data
- Interact with the database to change user data
- Handles user authentication

Database:
The database is a MongoDB database used to store user profile information, like username, password, level, and high score. It also stores the leaderboard by having a file for every player with their name and score. Lastly, it holds the location of photo spheres with their corresponding longitude and latitude.
The database has 3 different schemas:

- User schema
- Location schema
- Leaderboard schema

Frontend
The front end is used to build requests for the server and gather data for the back end to use in the game's operation. It provides UI for users to log in, play the game, view the leaderboard, and view their profile. The frontend also handles the game logic such as looping the game into rounds, calculating a users score, finding a location and displaying results.
Role:

- Sends request to server for user authentication, game data, and leaderboard
- Displays all the user interface
- Takes backend response and updates UI
- Perform game logic like random map locations and calculate score

6. Software design
User Interface Mockups:

1. Home
    a. Displays play game button, title, and images for design
2. About
    a. Consists of instructions on how to play the game
3. Leaderboard

    a. Shows users with the most points in a ranking system
4. Login
    a. Shows username and password fields
    b. Has a stay signed in button
    c. Has a login button
5. Signup
    a. Just like login, but with an extra field to rewrite your password for confirmation
6. Profile
    a. Shows username, high score, and level
    b. Has a logout button and a delete profile button

Database Schema:

1. User

username	String
password	String
high_score	Number
xp	Number

2. Location

path	String
long	Number
lat	Number

3. Leaderboard

username	String
score	Number

7. Coding Guidelines

JavaScript:https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/JavaScript

HTML:https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/HTML

CSS:https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/CSS