CS 411 Project Report

❖ Changes in Project Direction:

➢ In general, our project direction is not drastically changed from our project proposal created during stage 1. Specifically, our project continuously focuses on the innovative idea of combining crime rates and information (around restaurants) and restaurant information in order to provide users of our application a safe dining experience.

➢ However, some of the features initially included in our project proposal were removed/simplified due to various reasons. For instance, we chose to take out our community and group dining experience from our final application. This feature allows the users of our application to create an event at a designated restaurant, and invite friends or even strangers for a social dining experience. We decided to leave out this feature because of the difficulty to implement it and the limited time available for our team to complete the project. More importantly, because of the requirements for the ER Diagram (e.g. need to have at least 5 entities) in stage 2, we had to revise our database design, add entities, and remove the community feature due to the fact that our ER Diagram was becoming overly complicated to implement in the allowed time. If we are to continue developing our application, we would add the community feature to provide our users with social dining experiences.

❖ Achievements and Failures:

➢ Our project achieved almost all of the functionalities we planned. For instance, our application allows our users to search for restaurants with different filters such as location, rating, and crime data. In addition, users can add reviews and ratings to restaurants, search for dishes at restaurants, and modify restaurants' dishes.

- ➢ In terms of features we failed to implement, there are mainly three aspects.
    - ■ First of all, our search results are not organized well enough in terms of their visual representation. This is due to the fact that every member of our team is new to frontend, and had to learn frontend related skills from scratch.
    - ■ The second aspect is as aforementioned: we failed to implement the community and group dining feature due to time constraints.
    - ■ The third aspect is that it is better for our application to have a level of access functionality, where only users with access (e.g. restaurant owners) can modify the restaurant dishes. We did not implement this feature due to the fact that we want to easily demonstrate all of our features. In addition, we want to prioritize implementing the hard required functionalities within the limited given time.

- ❖ <u>Schema and Source of Data Changes</u>:
    - ➢ We did not modify our schema, except for only minor changes such as removing the neighborhood attribute of the Location entity, since we believe that this attribute is redundant to our application (i.e. Location/LocationId itself is sufficient).
    - ➢ In terms of the source of our data, we made several changes.
        - ■ Initially, our restaurant information and review information were from the Yelp Open Source Dataset and our crime rate information from the US

        Department of Justice Developer Resources, with other information such as User information self-generated. However, due to the fact that our initial crime data resource is overly expensive to access, we decided to self-generate our crime data with approval.

        - In addition, when we initially set up our Google Cloud Platform and imported all of our data as CSV, the size of our dataset was relatively small. When we found out that there is not enough data to perform queries on, we decided to augment our data: by importing more data from the Yelp Open Source Dataset and by self-generating more data. Eventually, we have enough data to work with.

- <u>ER Diagram and Table Implementation Changes</u>:
    - In terms of our ER Diagram, we did not change much since stage 2 besides removing the neighborhood attribute of the Location entity, since we believe that this attribute is redundant to our application (i.e. Location/LocationId itself is sufficient). However, we made many modifications to our ER Diagram during stage 2. First of all, we removed entities related to the community feature, such as the Community entity, and the Appointment and Invitation relationship. This, as aforementioned, is due to the fact that our ER Diagram was becoming overly complicated to implement in the allowed time. In addition, we added the Location entity and the Dish weak entity in order to make our application more useful, facilitate implementation, and satisfy the project requirements.
    - In terms of our table implementation, we did not change much since stage 2 besides removing the neighborhood attribute from the Location table.

- With the above explanation, we think that our finalized ER Diagram and Table design is the more suitable one compared to the original design.

- Functionalities Added or Removed:
    - In terms of the functionalities added in our final application, there are several ones.
        - First of all, we added a data visualization component in our final application, where users are able to see a visualized bar chart of the crime data (showing number of crimes of different crime types) related to a specific restaurant. This is because we want to facilitate our users' interpretation of the crime data of the restaurants they want to dine at, help them make better and faster decisions, and satisfy the extra credit requirements.
        - Moreover, we added the functionality to search for dishes and modify dishes for each restaurant. We added this functionality because we want to add more options for our users, and make our application more useful and robust. In addition, we added the Dish weak entity/table in stage 2. We want to make use of this table's data to improve our application.
    - There are also some functionalities that we chose to remove.
        - As aforementioned, we chose to remove our community and group dining experience from our final application due to the difficulty to implement it and the limited time available for our team to complete the project.
        - Furthermore, we chose to remove displaying independent restaurant information pages due to limited time constraints.

- <u>Usefulness of Advanced Database Programs</u>:
  - Our advanced database programs complement our application really well, in aspects such as providing functionalities, facilitating implementation, and adding robustness to our application.
    - First of all, the “GetRestaurantInfoWithCrime” stored procedure helped provide our application/users with the functionality to search for restaurants and related crime information with additional requirements such as the minimum star rating. In addition, the stored procedure contains a conditional structure, adding an input check functionality where if the input star rating is out of range, the procedure will generate an error message.
    - Moreover, the stored procedures such as “WithDishRestaurantInfo” facilitate our application implementation. Specifically, these stored procedures such as “WithDishRestaurantInfo” put advanced functionalities and group complex advanced SQL commands within a single procedure/function (in this case, “WithDishRestaurantInfo” contains SQL commands that return restaurant and crime information of restaurants that contains a specific dish/dishes that contains the input string in their names). When implementing our application using Flask, we can just call these stored procedures in the main file “app.py”, rather than writing out all of the complex SQL commands every time we have to use the functionalities contained in these stored procedures. As a result, the stored procedures significantly facilitated our implementation process.

        - Finally, advanced database programs such as triggers and transactions helped to make our application more robust. For instance, with our trigger “set_default_review_text”, we are able to set default reviews for users if they do not write comments (i.e. only enter the star rating) when adding a review to a certain restaurant. Furthermore, we have a transaction inside the “GetRestaurantInfoWithCrime” that only returns and commits the result if the input star rating requirement is in the required range (otherwise will generate an error message). These programs such as triggers and transactions improve our application so that it can handle many more situations and edge cases, making it more robust as a result.

- Technical Challenge:
    - Arthur Li: Since every member of our team is new to frontend development, we had difficulties setting up the frontend, making it look beautiful, and achieving our desired application design. Suggestion to future teams: start planning/learning frontend related matters early.
    - Chris Deng: Our team had trouble coming up with the required SQL commands due to the fact that our database programs have to satisfy all the requirements in the project description and, at the same time, be compatible with our application. Suggestion to future teams: come to office hours often and ask a lot of questions.
    - Justin Wang: We faced challenges when setting up the Google Cloud Database since it is our first time using it. We had difficulties figuring out some of the commands we had to use when setting up the database and implementing the

actual application. Suggestion to future teams: Go to the optional lecture about GCP, our team benefited from that lecture.

- ➢ Keran Wang: We had trouble finalizing our ER Diagram. This is because we had to satisfy the project requirements for the ER Diagram such as the “five entity” (your ER Diagrams must have at least five data entities) requirement and, at the same time, make our ER Diagram reasonable for implementation. Suggestion for future teams: go to office hours often, or stay after classes to ask questions, and keep the ER Diagram requirements in mind when creating your project proposal.

- ❖ <u>Other Things Changed</u>:
  - ➢ There are no major changes comparing the final project to the original proposal, except for those mentioned above. To reiterate those changes, they are:
    - ■ Removed Community, Invitation, and Appointment feature that allows users to have a social dining experience.
    - ■ Removed individual restaurant information pages.
    - ■ Added data visualization and dish access and modification functionalities.
  - ➢ The features are removed mostly because of the limited time constraints, as aforementioned. The features added are because of our desire to improve our application within the given time range.
- ❖ <u>Future Works and Improvements</u>:
  - ➢ In terms of future works that we can do to improve our application, there are several:
    - ■ First, we can add the originally planned Community, Invitation, and Appointment feature back to allow users to connect each other and dine

together through our application. We can do this by adding additional data entities and relationships such as the “Invitation” relationship between users.

    - In addition, we can add individual restaurant information pages to allow a more clear and concise view of each restaurant.
    - Finally, we can add a “level of access” feature to our application that only allows users with a certain level of access (e.g. restaurant owners) to modify the dishes. This will make our application more logical and authentic.

- Division of Labor and Teamwork:
  - In general, labor was roughly equally distributed.
    - Arthur Li: Focused on core parts in the almost all stages, such as:
      - Proposal in Stage 1
      - ER Diagram and Relational Schema in Stage 2
      - Advanced SQL programs in Stage 3
      - User registration and login system, part of designing SQL programs, perfection of UI, debugging of the entire application, data visualization, and project report in Stage 4
      - Create release and managed Git Repo (helped others set up Git repo and solve Git related issues) in every stage
    - Chris Deng: Focused on providing help on core parts in previous stages and implementation in Stage 4:
      - Helped write proposal in Stage 1

        - ER Diagram Explanation and Assumption in Stage 2
        - Indexing in Stage 3
        - Fixed Keran’s codes, designed SQL programs, and implemented many of the application’s features in Stage 4
    - Justin Wang: Focused on providing help in previous stages and implementation in Stage 4:
        - Helped write proposal in Stage 1
        - Description of relationship and cardinality and ER Diagram in Stage 2
        - Generating data for database in Stage 3
        - Implementation with Chris in Stage 4
    - Keran Wang: Focused on core parts and helped in previous stages and SQL commands in Stage 4:
        - Helped write proposal in Stage 1
        - ER Diagram in Stage 2
        - Indexing in Stage 3
        - SQL commands in Stage 4
- In terms of teamwork, most of our collaboration went well. However, there were certain communication issues amongst team members, which caused a disruption of harmony and normal work process within the team. Nonetheless, in the end, we were able to manage and overcome this issue and finish the project quite nicely.