Fundamentals Of Software Engineering Group - 1 Assignment - 4

Tuesday 17:30 - 19:00. Fall 21

Ian Cooper
Sayra Rodriguez
Jhaelon D Edwards Smith

September 21, 2021
Instructor R. Singh

Provide detailed project planning tasks document related to above requirements. Document must include:

Project scope

1. What the project is, why it's happening, and what it will achieve
   - The purpose of this project is to create a reservation system for a chain restaurant.
   - Providing customers an easier way to make reservation at their favorite restaurant.
   - No middle man, and no phone call needs to be made, customers can check availability and make the right choice from their smartphones
2. How the project will be completed (approach + phases + tasks)
   - Team communication
   - Architect
   - Technology and Roles
   - Deadlines
   - Documentation
   - Demo and Delivery
3. What will be produced (deliverables)
   - An iOS app will be available across the Apple App Store, however, it can be filtered to be accessible within certain cities or countries.
4. When it will be delivered (timeline + milestones)
   - Project will be divided into 3 parts:
     - Building the structure or UI(October)
     - Beta testing(November)
     - Delivery (December)

5. What it will cost (estimate + payment schedule)
    - The cost includes paying 2 engineers and 1 Architect to design and programing. As well as, the cost of Google cloud services which is pay as you go.
6. What is and isn't included (assumptions)
    - App will include Google Cloud service, encryption, reservation services, cancellation. App will not include Sign in with Apple, sign in with faceID or TouchID (feedback is required by customers to make decision).

Risk analysis and recommendations

A. Keep it simple
    - Application does not need to include overly complicated features, start simple with required features. Simplicity leads to a better design and easier debugging.

B. Test frequently
    - Every aspect of the application needs to be tested: elements included in frontend, backend, and other services. Testing helps us make better decisions; also help us either look for other solutions or revamp certain part of the application

C. Do not repeat
    - If a class is using the same method as other class, do not copy if the other class was not tested, this will lead to heavy debugging.

D. Use modular design pattern
    - Breaking down the application into modules that can be attached together to showcase a feature. Modular design is easier to debug and accessible across the application.

E. Document everything

- Documenting helps other developers understand what has been done, and what needs to be done if the work was not finished or needs to be debugged in future.

F. Predict, and manage all the risks that are being taken by the chosen solution to approach this problem. This solution could involve a lot of risks but most rewards, however, there could be another approach with less risks involved. These decisions needs to be validated and analyzed in the beginning so that everyone is on the same page and understand the risks involved.

Implementation plan (discuss development methodology, technologies, etc)

1. Agile is best choice for the development of this system. Agile helps the team deliver the highest quality product. It is extremely effective in team collaboration, as Architect would break the development into parts, set up meetings everyday (Agile SCRUM), and release part by part of the final product, easier to debug, and more transparency as what is being developed. More importantly, incremental releases could help collect feedbacks to improve the final product. Our team has chosen Agile SCRUM as the development methodology
2. Technologies:
    - iOS application using Swift
    - Google Cloud - real time database
    - Included Apple frameworks

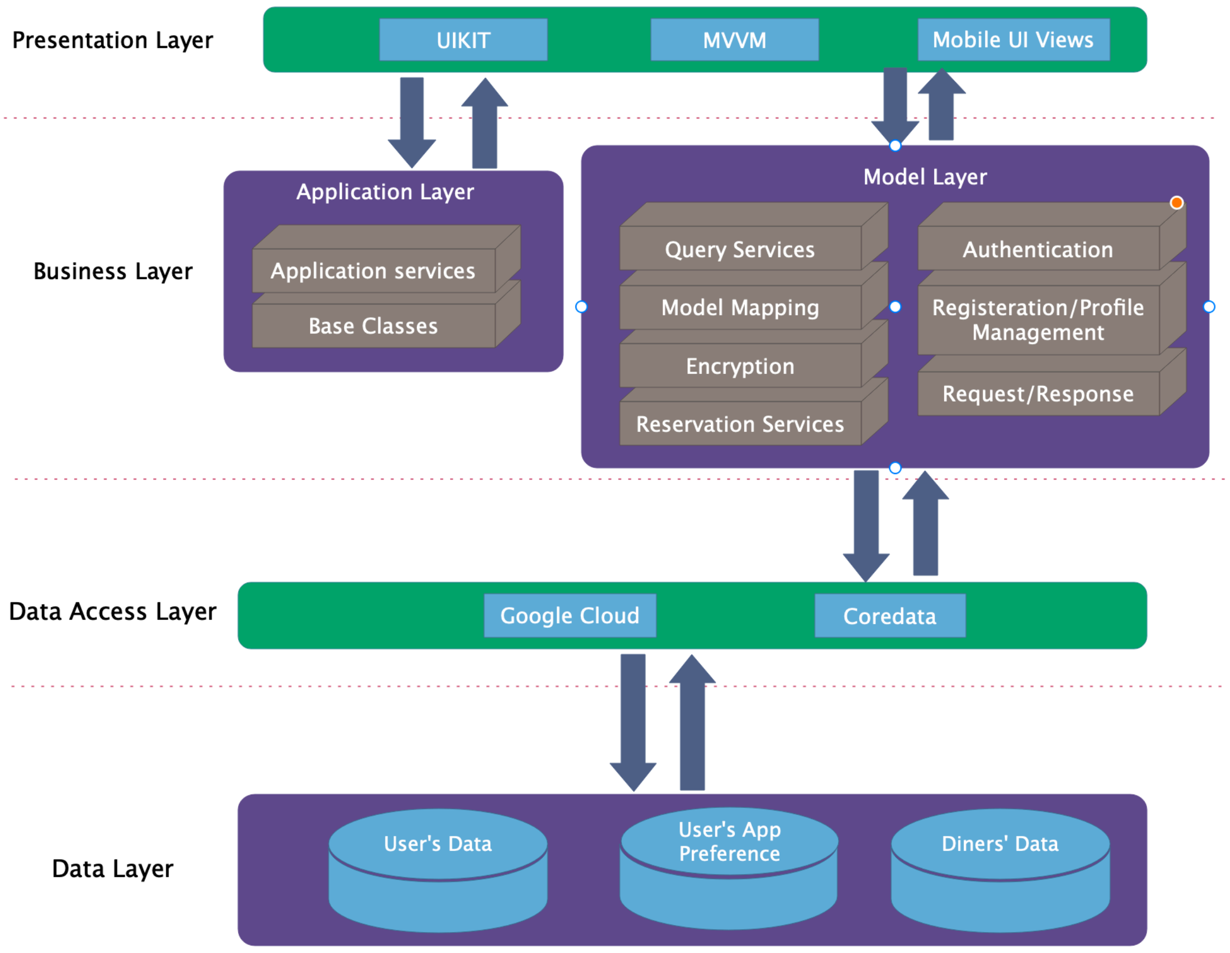
Project schedule (What will be delivered and when. Use weeks.)

1. October: the first four weeks would be invested on UI elements and overall structure of the Application.

2. November: the second four weeks would be dedicated to backend, and delivering a beta version of the app.

3. December: in the beginning of this month, the app will have final testing and prepare for shipping or demo.

Solution architecture (diagram & discuss each component/module)



1. Front-end: iOS application using Swift, and integrating with Google Cloud API for backend calls.

2. Back-end: Google Firebase Realtime Database to handle calls instantly, and more importantly, realtime database is going to be helpful with mutually exclusive part of the system, if tables are only available for one person, two people cannot reserve the same table configurations at the same time.

3. Framework: CryptoSwift is an encryption framework that is being used by companies like Bose for their transmissions between headphones and the phone. This framework will be used to encrypt calls/responses between the device and the server, as well as, encrypting user's information.

4. Others: for other requirements, iOS built in frameworks will be used, such as: Sign in with Apple(second option to register/login) provides users information that reside inside the phone eliminating the registration part of the app and credit card for no show charges, other provided frameworks are FaceID, TouchID, and Keychain(to store userLogin, if application was deleted, and installed later, keychain will provide user with login and password).

Testing plan

1. Analyze the product
   - Break down the product into testable parts. Since, not every part of the application can be tested in a dummy environment.
2. Design the Test Strategy
   - Making decision on who and how to approach our testing process. In our case, since the architect has higher understanding of app ecosystem should be overseeing the testing.
3. Define the Test Objectives
   - Start by listing all the functions and UI elements that needs to be tested.
   - Define targets and goals for features to be tested.

4. Define Test Criteria
   - Define both acceptable, and fail test cases. Understand what we are trying to achieve, and what would happen in case of failure.
5. Plan Test Environment
   - Certain part of the application can only be tested in a dummy environment. For example, the cloud API is one of them. We will design and decide which parts of the application should be in dummy environment and which parts are going to be tested in a simple assert function.
6. Schedule & Estimation
   - Testing needs to be done within a time frame. After, the architect defines the test objectives, team can request testing on their module or features as soon as they are done. Features will be tested, approved, and team will receive a response to move on to the next feature.
7. Determine Test Deliverables
   - There will be certain documents that will be provided before, during, and after testing; such as: test plans, test cases, and test designs which are included in every phase of our development lifecycle.

Quality measurement plan

1. Code quality
   - Easy to understand, bug free code is important for premium software. An application that could be picked up 5 years later by new developers and fully understood, is a quality app. Other measures would be maintainability, clarity, efficiency, and documents.
2. Performance

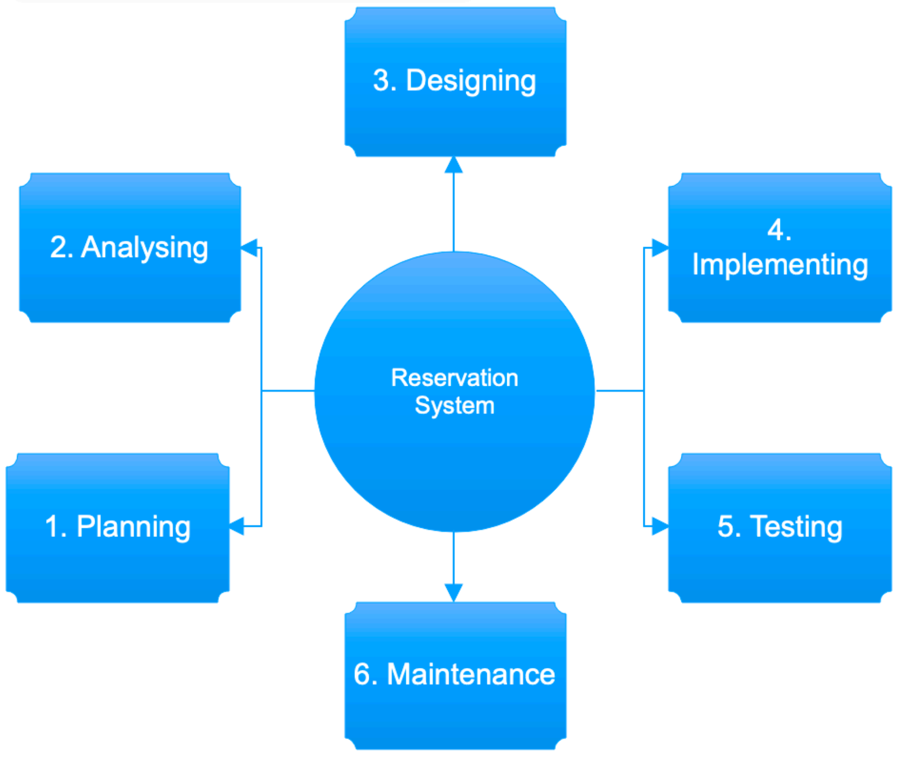
- It’s extremely import and priorities for us to make this application as responsive as possible. Reservations could be filled up, reserved by seconds. Having a responsive UI, along with fast and efficient backend could have a huge impact on our customer side.

3. Security
   - Making sure that no unauthorized changes could be done within the application is a key feature. Due to the fact that this application stores sensitive information, such as credit card. Security needs to be taken seriously. And for that reason, we are using top of the class encryption library (SwiftCrypto) to handle encryption and decryption.
4. Usability
   - Users are placed at the center of our priorities from performance to security and UI. Simple, easy to understand UI will have far more impact on users than a complex, busy UI that only certain users understand.
   - That’s why we are going with simple, and easy UI to reach users of all ages without the needs to explain every part of the applications.

Deployment plan (use diagram if needed)

1. Inputs
   - Gather all the requirements and criteria that are key affects.
   - Define roles and responsibility of the team
   - Assign tasks and resources
   - Communication tools between the team
   - Define debugging and testing methods
   - Designate a backup strategy
2. Outputs

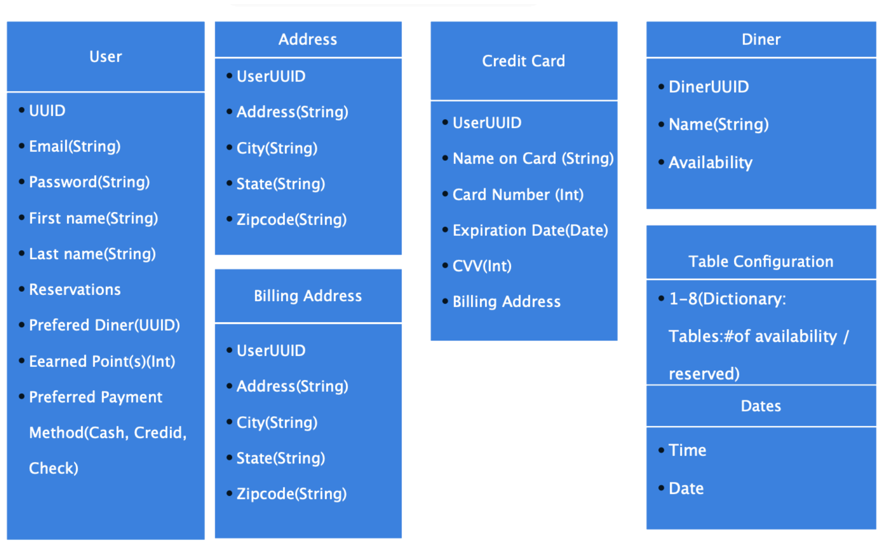
- Release plan: how, when, where this product is going to be released. Who is the distributor, what are the rules. What environment preparation needs to be done. i.e. Appstore has certain criteria that needs to be met to have the application published.
- Communication plan: choose and schedule time and dates for teams and all third parties such as, stakeholders, sponsors, and developers to be briefed about the progress and achievements.
- Progress tracking method: every successful software starts with this, if the team is not aware of progression throughout the app, their work will be heavily affected. i.e. one developer needs to implement localization, but before he/she can work on this, another developer needs to deliver a faster string compression algorithm.
- BC/DR plan: what needs to be done when something goes wrong. Having a contingency plan is extremely important when deploying a large scale application among presumably billion of users.

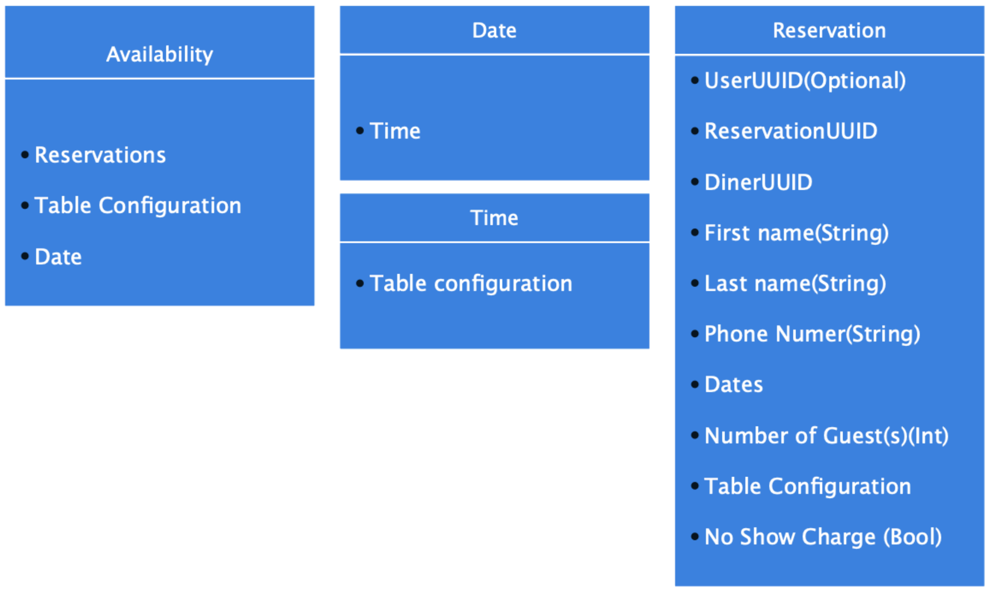


Maintenance plan (who will maintain and how customers log defects)

1. Corrective software maintenance
   - The team behind the application or assigned developer will be handling this type of maintenance. This type refers to necessary actions that need to be done once something goes wrong within the application and it could widespread or individual.
2. Perfective software maintenance
   - After publishing the applications, and collecting users' feedbacks. A new feature may come into our attention or a new issue may rise, that's when our team will start working on these topics.
3. Implementing a feedback and log system
   - Errors should be logged and backed up to Google Cloud to be visited by the team or assigned developer
   - Feedback system should be implemented within the application, so that users can report a bug, request a feature, or tell us how they feel about the application. And this could be done by having the backend collect feeds.

Database and UML Diagrams





Availability

- Reservations
- Table Configuration
- Date

Date

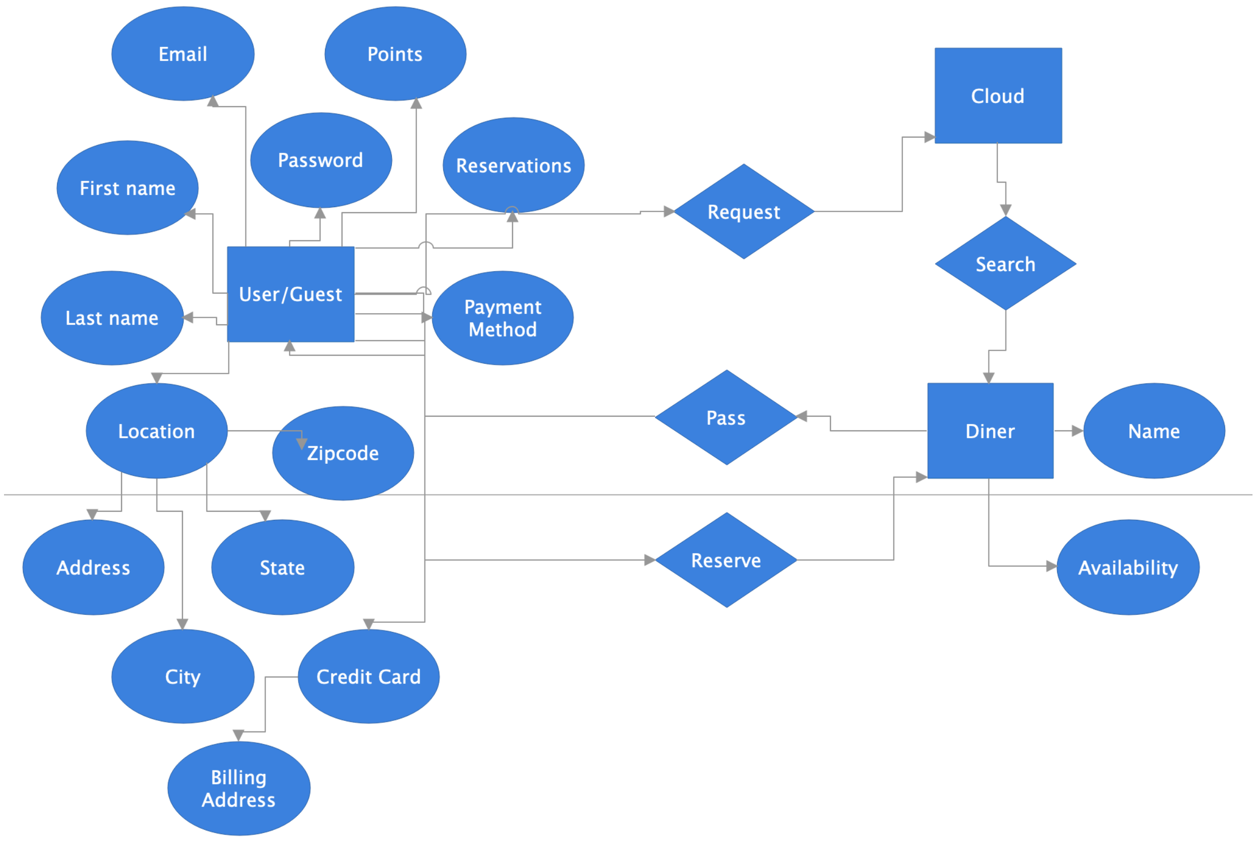
- Time

Time

- Table configuration

Reservation

- UserUUID(Optional)
- ReservationUUID
- DinerUUID
- First name(String)
- Last name(String)
- Phone Numer(String)
- Dates
- Number of Guest(s)(Int)
- Table Configuration
- No Show Charge (Bool)



Fill in this table, provide as much details possible:

Group Member Name	What is your contribution?	Discussion Notes
Ian Cooper	iOS Architect, Designer, Engineer	Discussed architect, model, risks and communications tool such as teams, or zoom. Describing testing and deployment plans. The back up or contingency solution for our application, if the iOS app could lead to a lot of problems, we could also switch to android or web application.
Sayra Rodriguez	Engineer	Focussing on the quality measures, code quality, how to break classes into modules. How to implement the feedback system: use email or create a custom system within the app. And where to store these information that are only accessible to our team or the company.
Jhaelon D Edwards Smith	Engineer	Gathering all the requirements , informations, and features that needs to be met by given deadline. Schedule time and date to meet expectations, look for the right tool to track our progress, i.e. GitHub provides a service that does that but look for other options as well. Helped in design of UML and database.