**DDD:**

In this project, I chose a Domain-driven design (DDD) in order to identifying the problem domains to create a software solution according to the domain inputs. It provides a structured approach to prioritize the domains and build software that closely reflects that understanding.

The following steps are listed below:

A)Various possible fields and events are gathered through brain storming and significant domains are listed. Link

B) A clear strategic design of the domains is provided coming from event storming

C) Core domain Chart

D)  bounded context to create boundary where each domain is applied.

A domain refers to the characteristics of the problem which the proposed software is to solve.

The Domain is what an organization does and the environment in which it operates. A software developer working for an organization must work within its domain. It is very important to divide models into logically separated subdomains according to their actual functionality. Subdomains allow different parts of the domain necessary to solve a particular problem to be identified more quickly.

This is a software development approach that focuses on understanding the domain of a problem to create a software solution. It emphasizes collaboration between technical and domain experts to iteratively refine a conceptual model that solves the problem.

**Core Domain:**

It is also necessary to be able to identify the Core domain. This is a very important aspect of the DDD approach. The Core domain is a subdomain that has primary importance for the organization. From a strategic point of view, the business should be distinguished by its Core domain. Most DDD projects focus precisely on the Core domain. The best developers and experts should be involved precisely in this subdomain. Most investments should be directed precisely here to achieve business advantage and gain the greatest profit.

**Supportive subdomain:**

If a specific aspect of the business is being modeled that is important but not part of the Core domain, then it belongs to the Supporting subdomain. The business creates a Supporting subdomain because it has a specialization.

**Generic Subdomain:**

If it does not have a specific purpose for the business and is required for the entire business as a whole, then it is called a Generic subdomain.

These types of subdomains are important for business success, but they are not of primary importance. It is the Core domain that must be implemented perfectly, as it provides a business advantage. This is the foundation for strategic design in the DDD approach.

**Event Storming:**

Event storming is a popular type of workshop that helps the software team align on what a solution should deliver. This is done without being distracted with the specifics of how it will be implemented. This means it may take longer for teams to start delivering source code. As a result of event storming, stakeholders end up speaking a ‘ubiquitous language’, such that there is little confusion around what the solution should do.

The workshop for event storming is a brainstorming exercise. In this session, all of the stakeholders of the solution work together to identify business events that correspond to domains. In the banking example, a business event may be a customer who applies for a new account. In the workshop the group will begin to identify the entity that triggered the event, the processes that must happen as a result, and any subsequent event that is triggered from the source event.

**Bounded Context:**

Bounded Context? Some weird name: what is it and why is it bounded?

To answer the question let’s come back to the problem we want to solve. We need to divide the model into comparably small pieces. The idea sounds simple but we need to do it in a way that they will be autonomous.

|  |  |  |
| --- | --- | --- |
| **Core Domain** | **Supportive Domain** | **Generic Domain** |
| 1. Task List Management  -- | 1. Notifications and alerts  - reminder  - real time data adaption  - message alert | 1. User Interface  - android Application  - web interface  -navigation |
|  |  |  |
| 2.  -  -user's info storage | 2. Rescheduling deadlines  -monitorization  -data update | 2. Platform Independency  -deployment  -setting  -configuration |
|  |  |  |
| 3. User Management | 3. Collaboration Context  - Shared task  -task assignments  -real time collaboration | 3. Security  -data security  -external authentication  -third party notification service |
| -profile creation  -login with with verification |  |  |
| 4. Activity Management | 4. Analysis and survey | 4. Integration |
| - add  -delete  -edit  -view  -update |  |  |

|  |  |
| --- | --- |
| 1. Core Domain | 1. Task List Management: maintain task, task category  2. Data Storage Management: task storage |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Core Domain** | **Supportive Domain** | **Generic Domain** |
| 1. Task List Management | 1. Notification context | 1. User Interface |
| 2. Data Storage Management | 2. Rescheduling deadlines | 2. Platform Independency |
| 3. User Management | 3. Setting and configuration | 3. Security |
| 4. Activity Management | 4. Analysis and survey | 4. Integration |