Technical University of Moldova
Inginerical department S.A.

Report

№: 6

AMOO

Subject: Statechart Diagrams. Domain Analysis. SOLID Principles

Author: Terman Emil FAF161
Prof: R. Melnic

Chisinau 2018

Tasks

- model the application using 3 **Statechart Diagrams**;
- perform the Domain analysis of your project;

Theory

Statechart diagrams

Statechart diagrams describe different states of a component in a system. The states are specific to a component of a system. Such a diagram describes a state machine. State machine can be defined as a machine which defines different states which are controlled by external or internal events.

Here are the main purposes of using *Statechart Diagrams*:
- to model the dynamic aspect of a system;
- to describe different states of an object during its life time;
- define a state machine to model the states of an object;
- to model the life time of a reactive system;

Domain Analysis

Domain analysis is the process of analyzing related software systems in a domain to find their common and variable parts. It is a model of wider business context for the system. Domain analysis produces domain models using methodologies such as domain specific languages, feature tables, facet tables, facet templates and generic architectures, which describe all of the systems in a domain.

SOLID Principles

The term SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible and maintainable.
- **S**ingle responsibility principle: a class should have only a single responsibility (i.e. changes to only one part of the software's specification should be able to affect the specification of the class);
- **O**pen for extensions and closed for modification;
- **L**iskov substitution principle: ”objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program”;
- **I**nterface segregation principle: ”many client-specific interfaces are better than one generalpurpose interface”;
- **D**ependency inversion principle: one should ”depend upon abstractions, **not** concretions”;

GRASP Principles

The different patterns and principles used in GRASP are controller, creator, indirection, information expert, high cohesion, low coupling, polymorphism, protected variations and pure fabrication. All these patterns answer some software problems, and these problems are common to almost every software development project.

These techniques have not been invented to create new ways of working, but to better document and standardize old, tried-and-tested programming principles in object-oriented design.

Statechart diagrams

In *Figure 1* is represented the Statechart of creating a new event. The user can create a Public, Friends Only or private event. When the event is private, the event can be joined through a link. When the user submits the request, few things happen at the same time.
- the user is shown that he has successfully created an event;
- if it's Public or Friend Only event, the Invites are sent and the event is posted on the timeline;
- the submited event content is checked by some algorithms. If it detects that something is inappropiate, the event is hidden and a notification to an admin is sent at the same time. When the admin checks the notification, he double checks if the event is indeed inappropiate. If he decides that it's fine, the event is unhidden, otherwise, the event is deleted and a report is registred on the event owner. The user then receives a warning.

Figure 2 is about adding a new photo. After the user choses a valid file, he can add a comment, tag some people or add a location in ”parallel”. After the image is submited, the following actions start in parallel:
- the comment and the image is queued for validation;
- the action is posted on the user’s timeline;
- tag approval is sent asynchronously to tagged users;

After these, the user is redirectioned to his newly posted photo.

In *Figure 3* is shown the statechart of a friend request. After the user submits a friend request, the server double checks if requested user exists. If it doesn’t, it most likely means that it’s an external request. This will register an Attack report on the request owner and an adim is notified about this. If he does exist, the platform runs through the following states:
- add the target user to the owner’s friend requests list;
- add the owner to the target’s friend requests list;
- send notification to taget;

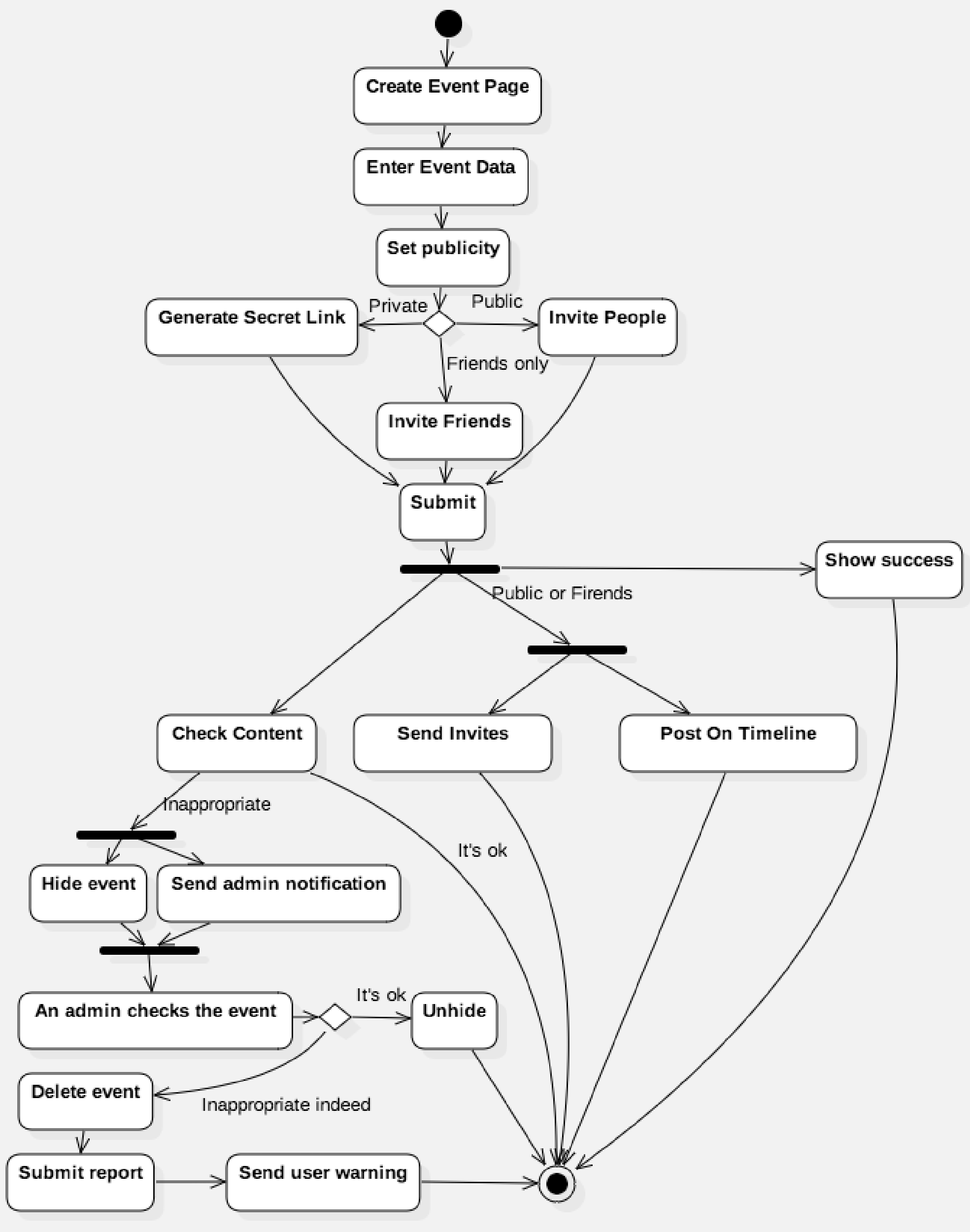
After all these actions are completed, a success message is displayed.

Domain analysis

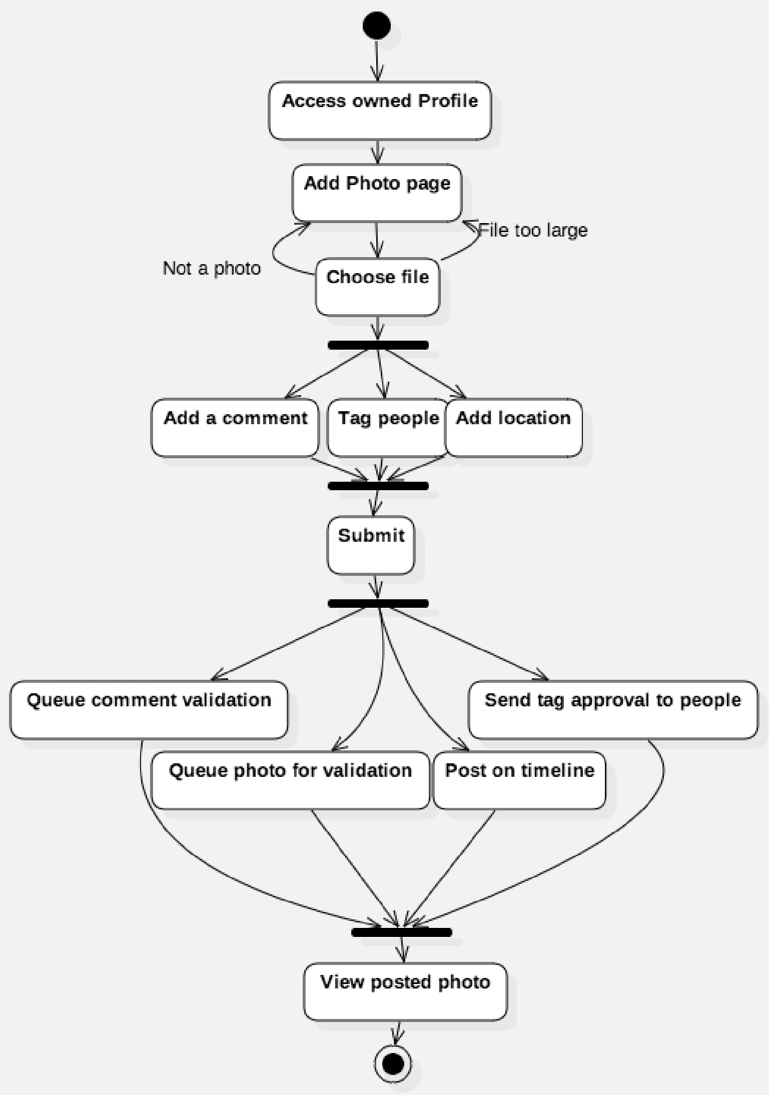
- this platform is from the social networking **domain**. People are continously looking for a faster, intuitive and safier app, so I think, my application’s main advantage is the full comunication privacy. Another strong point is that this project will be available as open source;
- *The importance of this theme*: a user’s privacy is often secretly ignored, or at least, the user isn’t 100% sure his chats are private. So, it would be a good idea to make the project open source so that other programmers can confirm my project’s privatness;
- **Twiter** and **Instagram** are **similar applications**, but the user’s private content is used for advertistment targeting and other nasty things. Instagram is more media based platform, whereas Twiter, it’s more focused on comunication;
- *The purpose and objectives*: This app is supposed to be open source, meaning that there will be many comunity contributions, which will greatly affect the popularity.

Conclusion

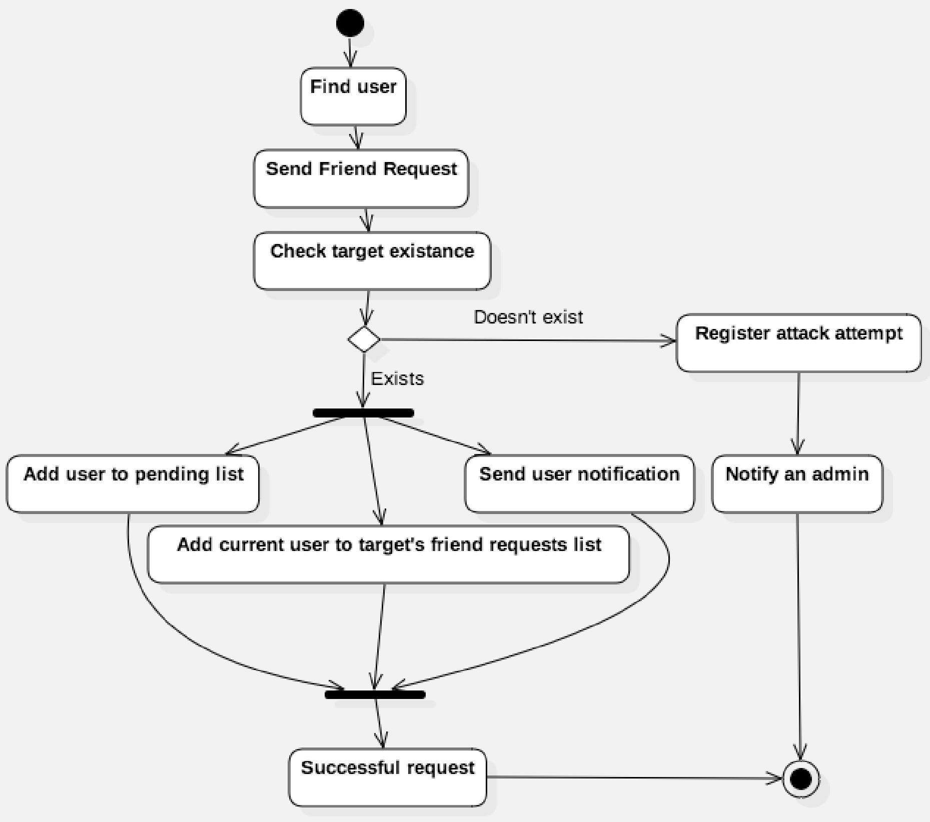
In this laboratory work I have learned how Statechart diagrams can be useful in real life production. It is another great way of explaining to non programmers what are the main ideas of a project. During this lab I took a quick glimpse to *Domain analysis*. While making this analysis, I came to the conclusion that my app needs more features to have a chance to survive on the market. So I came up with the idea of making it an open source project. Which acutaly may work out...



Img 1: Create an event



Img 2: Add new photo



Img 3: Friend request