**Your Name** <your-ju-id@student.ju.se>,  
**Your Name** <your-ju-id@student.ju.se>

A Project Work in *Internet and Cloud Services*

Jönköping University 2018

Your platform title here

*In this template, all italic text should be removed and replaced with your own text (which should not be italic); the italic text is just a placeholder letting you know what to write there.*

*On the cover page, change to your own platform title, your own names and your own JU email address.*

*You have a lot of freedom when it comes to writing this report. You do not have to use any part of this template, but the report you write should in the end somehow (in a good way) provide the same information as indicated in this template. Most students trying to do it in their own way usually fail, so if you try that, be sure to know what you are doing!*

*This page should be removed.*

Table of Contents

[Introduction 3](#_Toc528136262)

[Architecture 4](#_Toc528136263)

[Resources 5](#_Toc528136264)

[Backend Application 6](#_Toc528136265)

[REST API 7](#_Toc528136266)

# Introduction

??? is a mobile single-threaded forum where people can make a post about anything or upload a picture. The post will then pop up for all the users who live in the same city. They can then comment and like/dislike this post. This app is similar to another popular forum app that is called “Jodel”. The difference here is that in Jodel, every user is anonymous, no name and no profile. In ??? users will have their own username and profile where they can have a picture and a textbox where they can write whatever they want for people who visits their profile to see. The idea is to make the experience more personal, even though you can still choose to be anonymous by having an irrelevant name and profile picture, you will still be recognized and remembered by other users from your username.

*Introduce your platform. Write text that* ***indirectly*** *answers questions like:*

* *Why does the platform exist/what is the problem it solves?*
* *How does the platform solve the problem?*
* *Why is your platform solution better than existing solutions (if any exists)?*
* *How will end users use the platform?*
* *...*

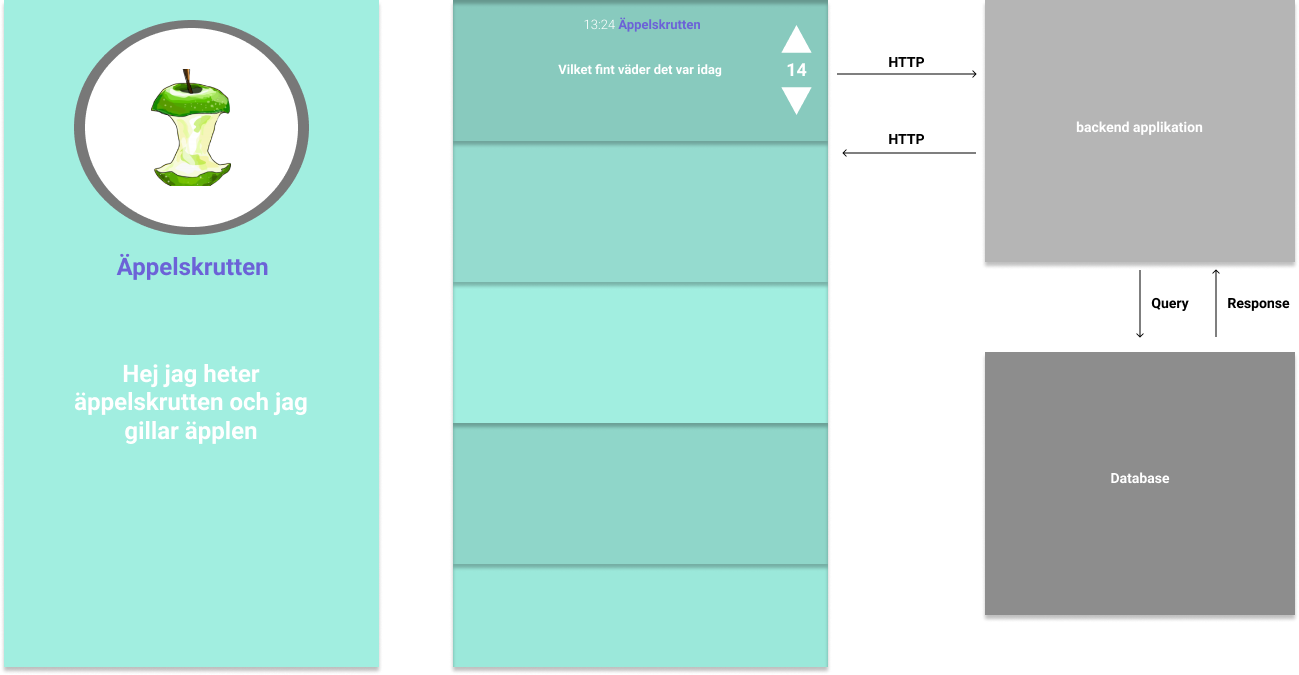
***Indirectly*** *means that you should avoid using the words* ***problem*** *and* ***solution****. For example, instead of writing "A problem with laptops is that they run on batteries which don't last forever" you can write "Batteries in laptop don't last forever, so often people can't use them as much as they want to".*

*After having read this chapter, those that have never heard of the platform before should have a good understanding of what it is. If they would like to learn how it has been implemented, they just need to continue reading the rest of the report.*

*If possibly, provide pictures/figures of some kind. Maybe a use-case diagram?*

*This chapter can to some extent be seen as a pitch text: imagine the reader is an investor, and you should convince the reader that your platform solves an existing problem in an excellent way and that it is worth investing money in it.*

# Architecture



*The application consists of three different components.*

***The frontend application*** *handles the visual interface.*

***The backend application*** *handles the CRUD operations. Such as When a user is making a comment, liking or disliking a post, changing profile picture or profile description, creating a post or deleting a post or creating account/logging in. The backend will send requests to the database accordingly.*

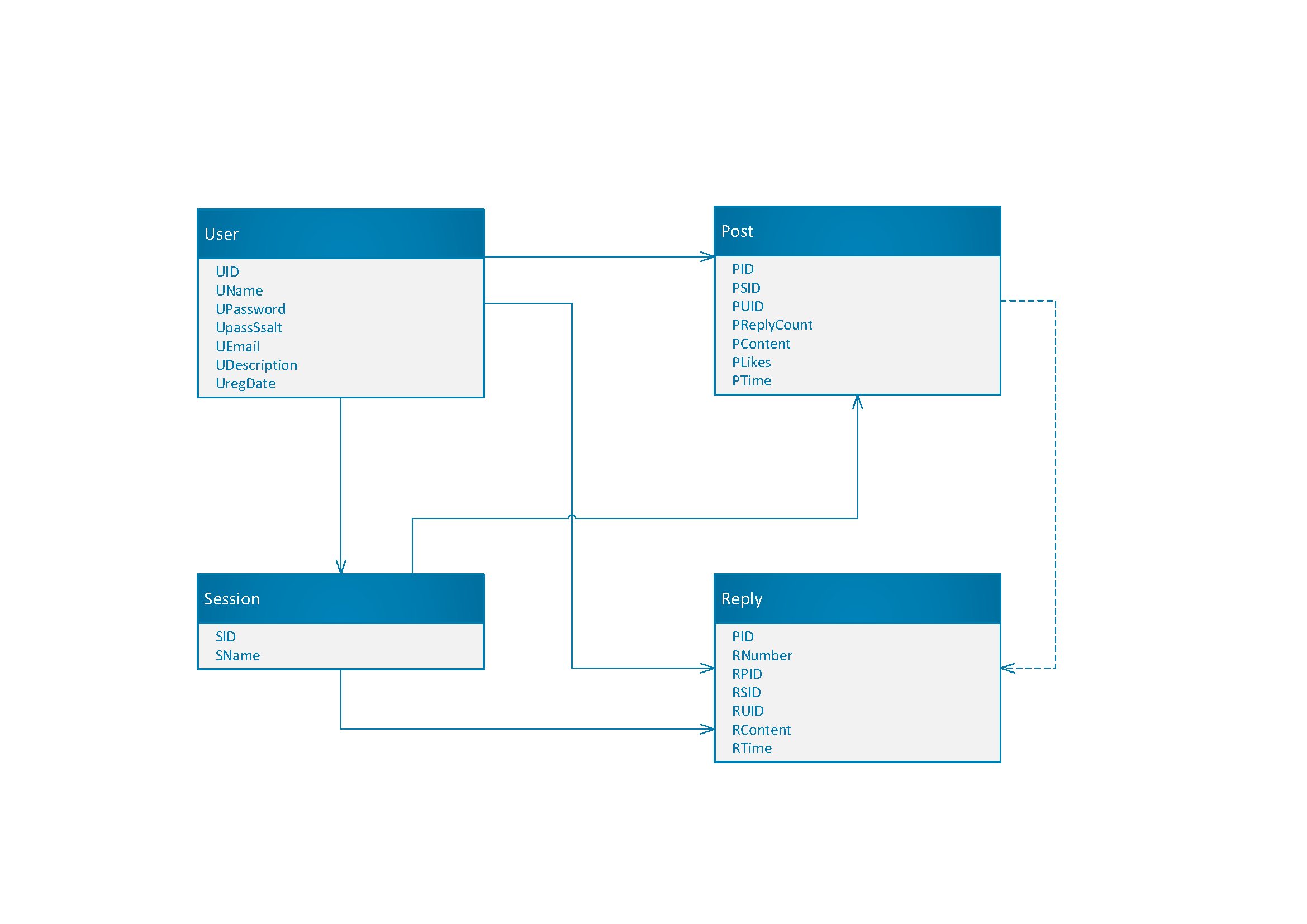
***The database*** *stores all posts, comments, account values and pictures.*

*Give an overview of the platform. Which components does it consist of (backend application, frontend applications, frontend devices, databases, etc.)? Visualize this using a figure and show how the different parts make use of/communicate with each other.*

*For each component, provide a sub-chapter with more information about how that component works. You will not implement any frontend application in this course, so only give a brief description of how these works. The backend application should be described in detail in the chapter Backend Application, so only describe that one briefly here as well.*

*After having read this chapter, the reader should have a broad (but shallow) understanding of the entire platform.*

# Resources



There are four different resources on the database as User, Post, Reply, and Session.

* The ‘**User**’ entity contains everything needed to have many unique users. The attributes would be User ID, User Name, Password, Salt for the password, email, description for the profile, and the registration date.
* The ‘**Post**’ entity contains everything needed to have many unique posts. The entity has a reply count to know the amount of replies below it. The attributes would be Post ID, Session ID, User ID, Reply count, Content, Likes, and the time the post was created.
* The ‘**Reply**’ entity contains everything needed to have unique replies connected to a post. The entity need a post to be created. The attributes would be Post ID, Reply Number (position in an array for example), Reply ID, Session ID, User ID, Content, and the time the reply was created.
* The ‘**Session**’ entity will be used to know the current session. With the attributes Session ID and Session Name.

*Describe the resources in detail. What attributes do they consist of? Showing an ER diagram might be a good way to visualize the resources?*

*After having read this chapter, the reader should know how the data stored on the platform is structured. If the reader is a new programmer that should start working on the platform, she should now know what she needs to know if she wants to change the resources or add more type of resources (e.g. know how to add a new table to the database with a relation to an existing table in the database).*

# Backend Application

*Describe how you've implemented the backend application. Which language have you used? How has the code been structured? How does one start the application? Etc.*

*After having read this chapter, the reader should know how the backend application has been implemented. If the reader wants to add a new type of resource to the platform and implement CRUD operations for that one, the reader should now know precisely which files that should be created/extended to contain the new code dealing with the new resource.*

# REST API

*This chapter should contain the specification for the REST API on your platform. It should contain enough information for a new programmer to start using the REST API without reading through the code on the backend application implementing the REST API.*

*If you prefer, feel free to write this chapter in a separate document, and just provide a reference to that document in this chapter.*

*You are recommended to look at how others have specified how their REST APIs work and then choose a way to describe your own REST API that you think is good. You can for example look at:*

* *Facebook:* [*https://developers.facebook.com/docs/graph-api/reference/v3.2/album*](https://developers.facebook.com/docs/graph-api/reference/v3.2/album)
* *Google Calendar:* [*https://developers.google.com/calendar/v3/reference/calendars*](https://developers.google.com/calendar/v3/reference/calendars)
* *GitHub Project:* [*https://developer.github.com/v3/projects/*](https://developer.github.com/v3/projects/)