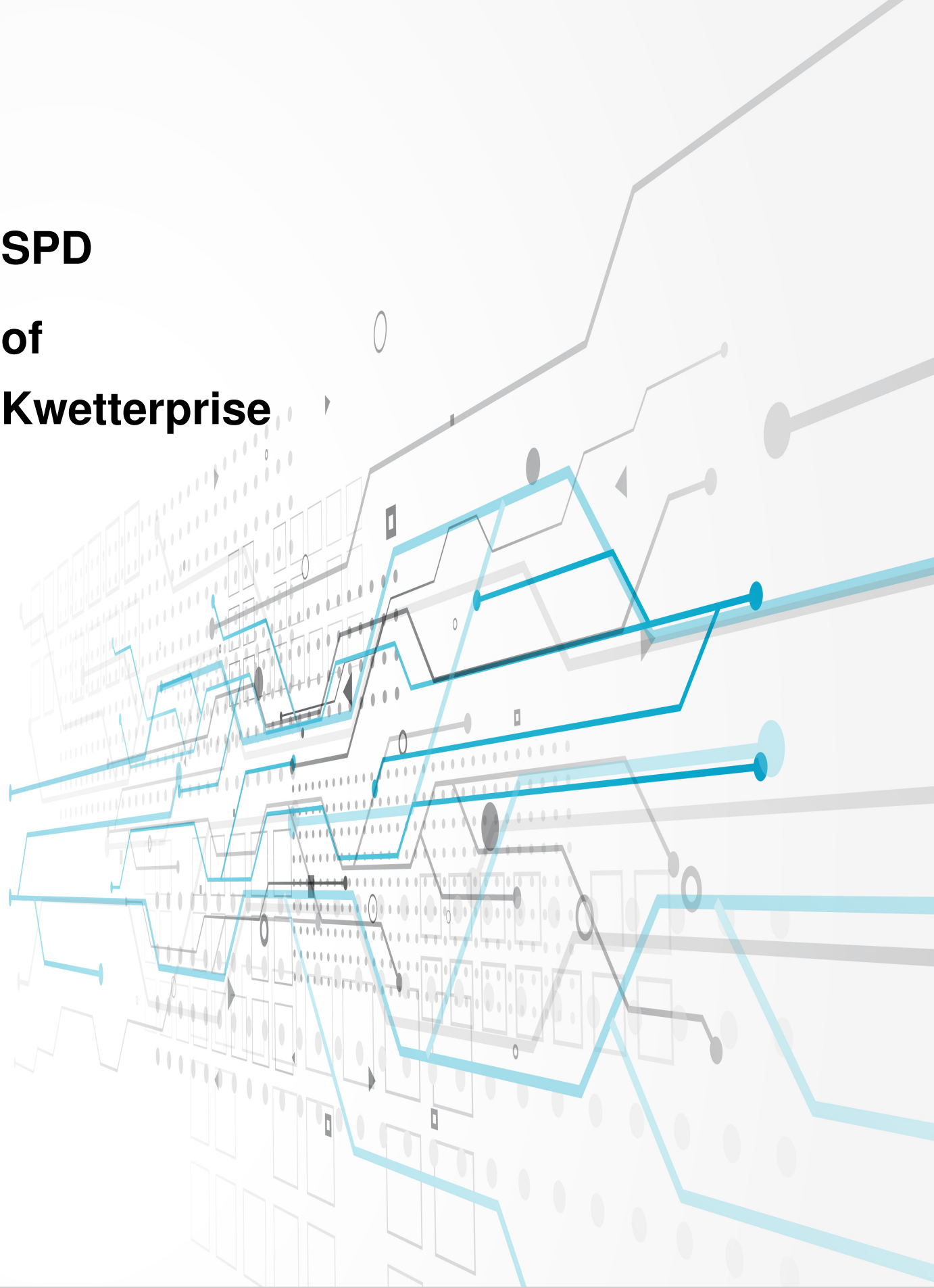
SPD

of

Kwetterprise



Date: 2020-04-15
Author(s): Tim van den Essen
Status: Concept

Course: S6
Student number: 3118525

Document history

Rel.	Date	Status	Changes
R01	2020-04-15	Concept	First version

Distribution list

Name	R01	R02	R03
Kwetterprise			
Tim van den Essen	A		
Fontys			
Erik van der Schriek	X		
Frank Coenen	X		

Document

Text marking

Marked text	Text needs to be changed or completed.
Marked text	Text has changed compared to the previous release.
Marked section	Section headers that are intended for review.

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

This document describes the design of Kwetter.

1.1. Definitions and abbreviations

RSD	Requirements Document
SPD	Specification Document
EDD	Engineering Design Document
API	Application Programming Interface
JSON	JavaScript Object Notation

2. Requirements

This capter contains functional and non-functional requirements for Kwetterprise. Functional requirements describe what the system should do and non-functional requirements describe how the system works. Note that this list is incomplete.

2.1. Functional requirements

2.1.1. Tweeting

`[F_TWEET_POST]` A user can post a tweet. Must

`[F_TWEET_GET_FROM_USER]` The tweets from a user can be retrieved. Must

`[F_TWEET_REALTIME]` An overview of real-time most recent tweets can be viewed. Must

2.1.1.1. Replying

`[F_REPLY]` A user can reply to an existing tweet. Could

2.1.2. Reacting

`[F_REACT]` A user can react to a tweet. Should

`[F_REACT_SHOW]` A tweet shows the amount of reactions it has received. Should

2.1.3. Following

`[F_FOLLOW]` A user can follow another user. Must

`[F_FOLLOW_DASH]` A user can view a dashboard with recent tweets by followed users. Must

2.1.4. Users & roles

`[F_REGISTER]` A user can register themselves through a register page. Must

`[F_LOGIN]` A user can log in. Must

`[F_LOGOUT]` A user can logout. Must

`[F_MODERATION_DELTWEET]` A moderator can remove tweets. Should

`[F_MODERATION_BAN]` A moderator can ban users. Should

`[F_ADMIN_ASSIGN_ROLES]` An administrator can change the role of a user. Should

2.2. Non-functional requirements

`[APIG_REQ]` The API gateway must be able to handle 500 requests per minute. Should

3. Accounts and Rights

Within Kwetterprise, three roles exist:

1. **User**
   A user can post and read tweets.
2. **Moderator**
   A moderator can delete tweets and ban users.
3. **Administrator**
   An administrator is a moderator with the additional right of changing user's roles.

When an account is created, the default role is "User". Table 3.1 describes which actions can be performed by which roles.

Table 3.1: Roles and rights.

Right	Roles		
	User	Moderator	Admin
Read Tweets	X	X	X
Post Tweets	X	X	X
Follow another user	X	X	X
View other's followers	X	X	X
React to a tweet	X	X	X
Delete tweets	X*	X	X
Ban users		X	X
Change role of user/moderator to user/moderator			X
Change role of user/moderator to administartor			X

*Only their own tweets.

4. Mockups

This chapter contains mockups of Kwetterprise. Note that Chapter 2 has a better overview of requirements and their priority.

4.1. Home page

Figure 4.1 depicts a mockup of the home page. Table 4.1 describes the functionality on the page.

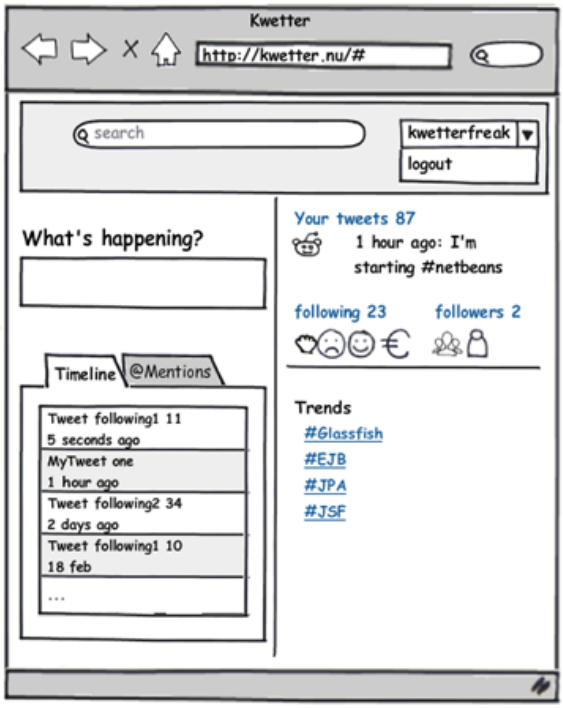


Figure 4.1: A mockup of the home page.

Table 4.1: Home page.

Action	Description	Priority
Profile details	User details: bio, location, website Bio is limited to 160 characters	Must
Own tweets	10 most recent tweets by this user	Must
Followers	Other users following this user. By clicking followers in S3, a list of followers appears in S1	Must
Following	Other users followed by this user. After clicking one of the links in S4, information of that user wil appear in S4	Must
Profile picture and name	Profile picture and name of the user	Should
Heart	Users can add a heart symbol to the tweet of another user. A user can give at most one heart symbol to each tweet.	Could

4.2. Profile page

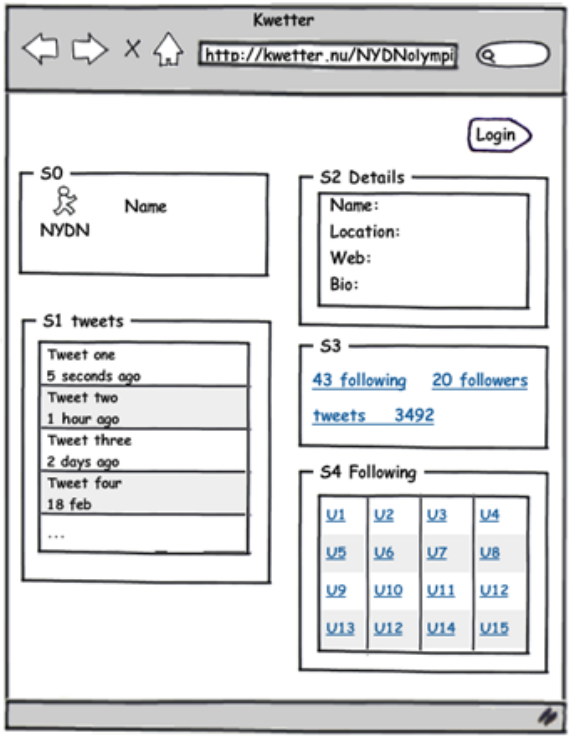


Figure 4.2: A mockup of the profile page.

Table 4.2: Profile

Function	Description	Priority
Search	Search within all tweets. Results will be shown in the timeline tab.	Must
What's happening?	Textbox to enter your tweet with a maximum of 140 characters. The tweet will be shown immediately under 'Your tweets'.	Must
Timeline	Timeline of tweets from the user and from other users being followed.	Must
Following & followers	After clicking on either the following or followers link, the profile page appears.	Must
Logout	After clicking logout the profile page appears. It is then possible to login as a different user.	Must
Login	After clicking login and providing credentials, the starting page appears filled with tweets and other information for the user that is logged in.	Must
@Mentions	Tweets mentioning the user appear in this tab. It is possible to mention another user by adding @username in the tweet.	Should
Trends	Clicking on one of the trends results in list of tweets in the timeline tab. These tweets are at most one week old and contain a hashtag followed by the selected trend.	Should

4.3. Registration

Registration form for new users.

4.4. Moderation

A page for users with the role of moderator. Moderators can see a list of users and their roles. They can remove users who spread insulting tweets or use swear words.

4.5. Administration

An administrator can add or remove accounts of other users and may change the roles of other users.

5. API Gateway

This chapter discusses the specifications of the API gateway.

5.1. Definitions

- **Service**
  A service refers to the definition of the API of a microservice. An example is "`email-service`". This shows that a service exists with that name.
- **Service identifier**
  Multiple instances of one service can exist. Each instance is identified by its own GUID. A generated GUID is practically unique.
- **Endpoint**
  The endpoint

5.2. Routing

Requests are made to `http://api-hostname/api/service-name/endpoint`.

The API gateway re-routes this request to the approperiate server. If no server is running, a HTTP status code of 503 (Unavailable) is returned.

Query parameters (such as `/endpoint?value=true`) and POST bodies are forwarded to the service as well.

5.3. Service discovery

5.3.1. Service requirements

Certain requirements are put on services which register themselves at the API gateway.

1. **Registering**
   A client registers themselves by sending a request POST to `\register`. The POST request must contain the following body:

   ```
   {
       "ServiceName": "[The name of the service]",
       "Guid": "[The generated GUID]",
       "BaseUrl": "[The base url this service is reachable at]"
   }
   ```

   It is OK—and even encouraged—to send duplicate register requests. This is useful when the API gateway has restarted (and lost its services) but the service is still running.
2. **Ping**
   The service must have a `/status` endpoint which returns the status of the service. This status must be:
   - `200` when the service is running OK.
   - `504` when the service is (partly) not available due to issues.

   If this endpoint is not reachable, the service is assumed to be irreparably crashed and is removed from the service discovery. The service must register itself again.
3. **Unregistering**
   Unregistering is optional but preferred. If the service stops without unregistering it will eventually be removed because `/status` is unreachable. A service can unregister itself by sending a POST request to `/unregister` with the following body:

   ```
   {
       "Guid": "[The GUID of this service]"
   }
   ```