MyTwitter

Course - CSC 8542

Instructor - Professor Kristin Obermyer

Team members - James Keys, Sri Sowmya Moturu and Matthew Janny

Agenda

1. Overview
2. Requirements
3. Marketecture
4. Risks
5. Design Views
6. Prototype
7. Retrospection

Overview

Twitter has two basic modes :

- Home and Latest

Need for some customization for giving a better experience for the Twitter user

- Developing a software for customizable Twitter timeline
- User preferences to be taken into account while customizing their timeline
- Adjusting recency bias as per user preferences
- Adjusting following bias as per user preferences

Customizing User Preferences

Users interact with the app in different ways and want an app which fits their specific needs accordingly:

- Follow weighting
- Time weighting
- Virality weighting
- “Like” toggling for followings

PLANNING

Functional Requirements

As a user, I want to be able to create a profile with the application so that my account preferences save from the time of last use

As a user, I want an application with a user interface that shows a timeline of tweets so that I can read tweets directly from the app

As a user, I want to be able to select whether or not my timeline shows the liked tweets of people I follow so that I can see the tweets that are most relevant to me.

As a user, I want to be shown the most-interacted tweets over a specified amount of time so that I can catch up on important events that I missed.

As a user, I want to be able to adjust the recency bias of the order in which tweets are shown on my timeline so that I can focus more on tweets from a specific time.

Dig a shallow trench

Non Functional Requirements

Usability:As a user, I want to be able to see tweets as old as 7 days in my timeline so that I can catch up on anything important I may have missed in the last week.

Performance :

As a solution architect, I want the database to be robust enough to handle the user data of nearly 100,000 users

Availability:As a solution architect, I want the application to have a 99.99% availability so that the user can have a good user experience

Availability and usability:As a user, I want timeline page containing all the tweets to be loaded in less than 2 seconds so that I can have less wait time to go through the tweets that are tweeted by the accounts I follow

Reliability and Maintainability:The mean time to repair (MTTR) functionality should be less than 4 hours and it should not have any impact on other functionalities or modules. In case any downtime occurring within the functionality, the pages related to it should specify a message stating the specific downtime so that user gets to know when to access it again

Architectural Mechanisms

Availability

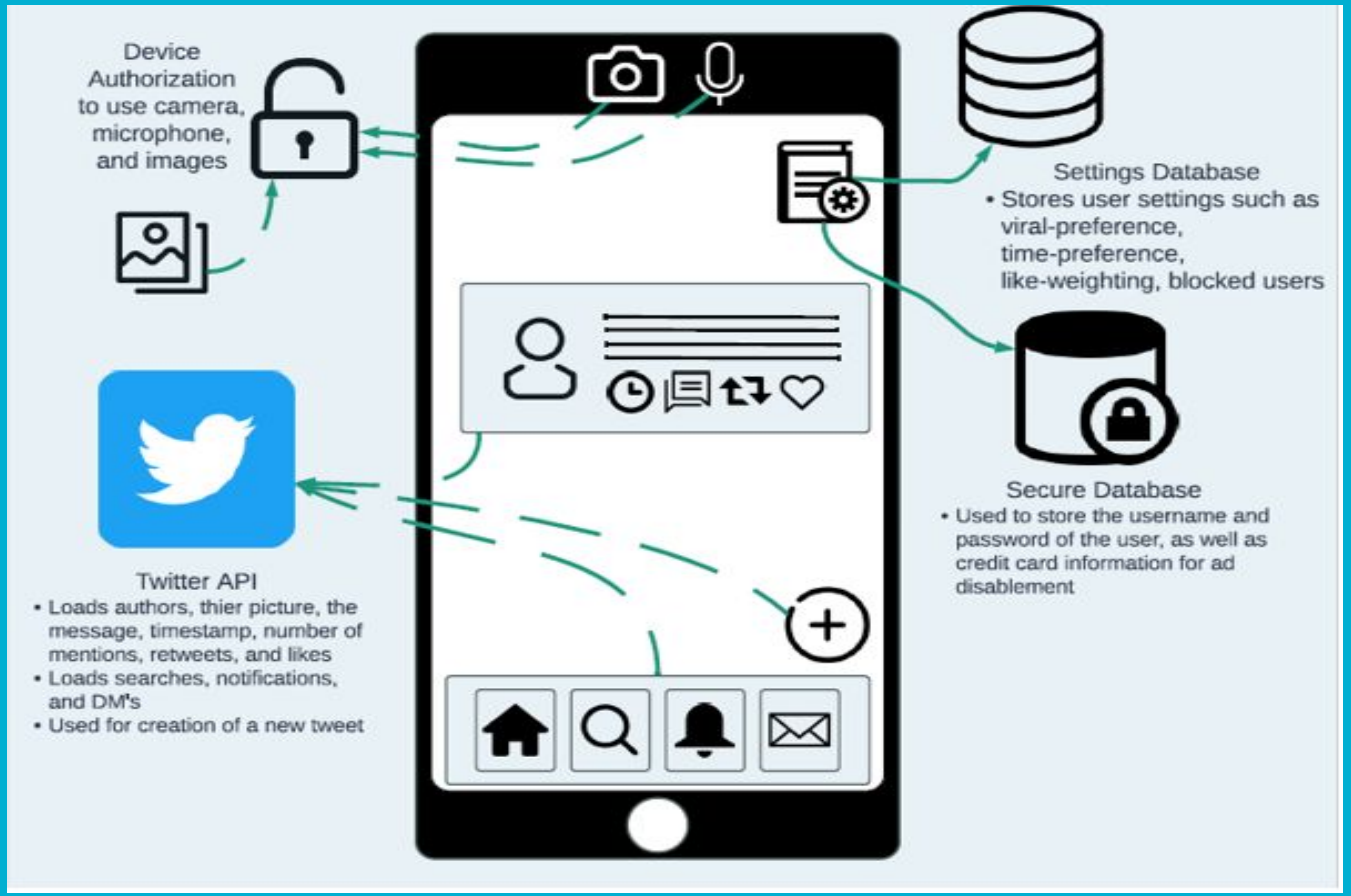
Communication

Workflow

Security

Persistence

Marketecture of MyTwitter



Estimated Risks

Technical Risks

Business Risk

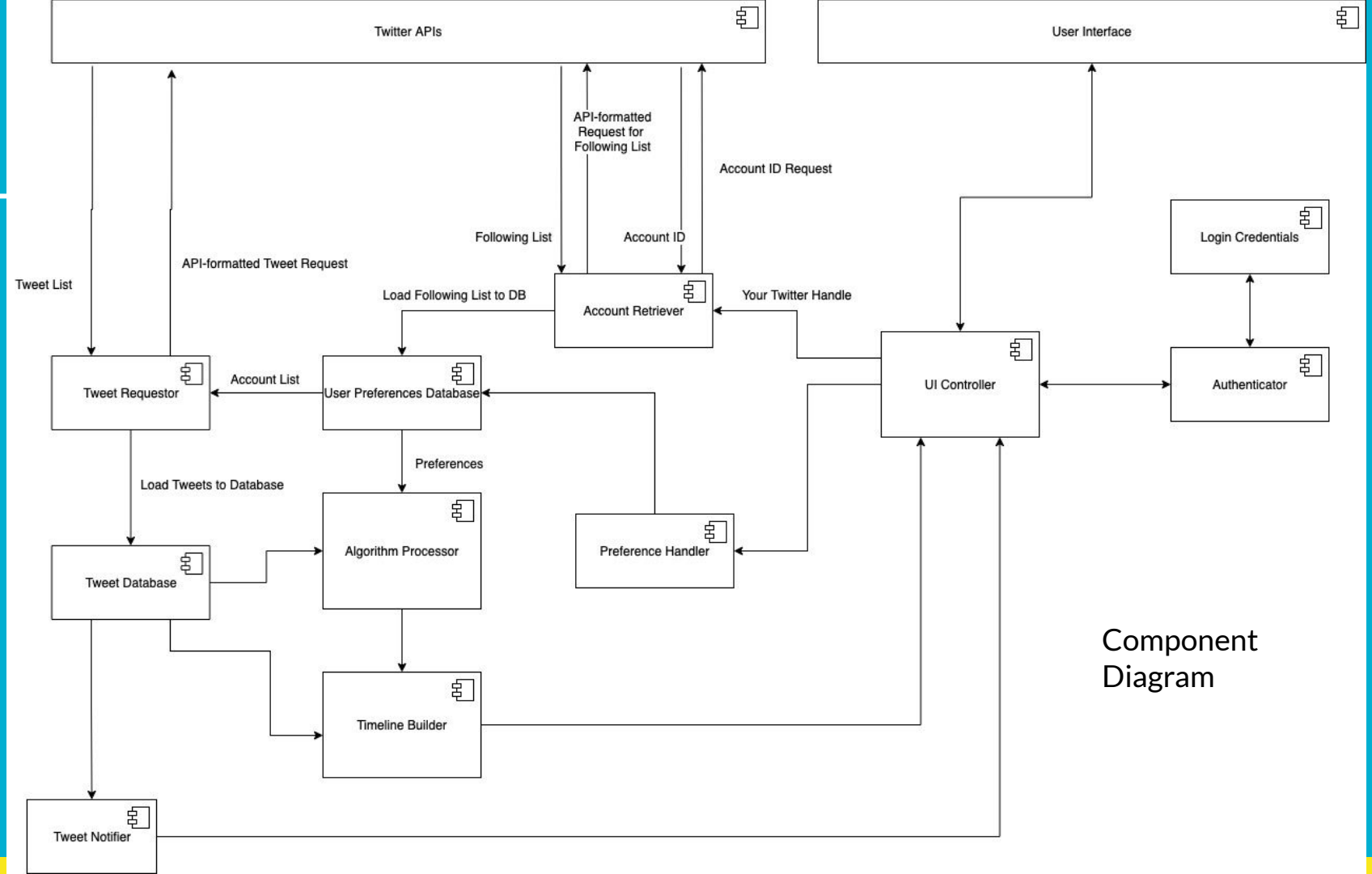
Human Risk

DESIGN

What did we design?

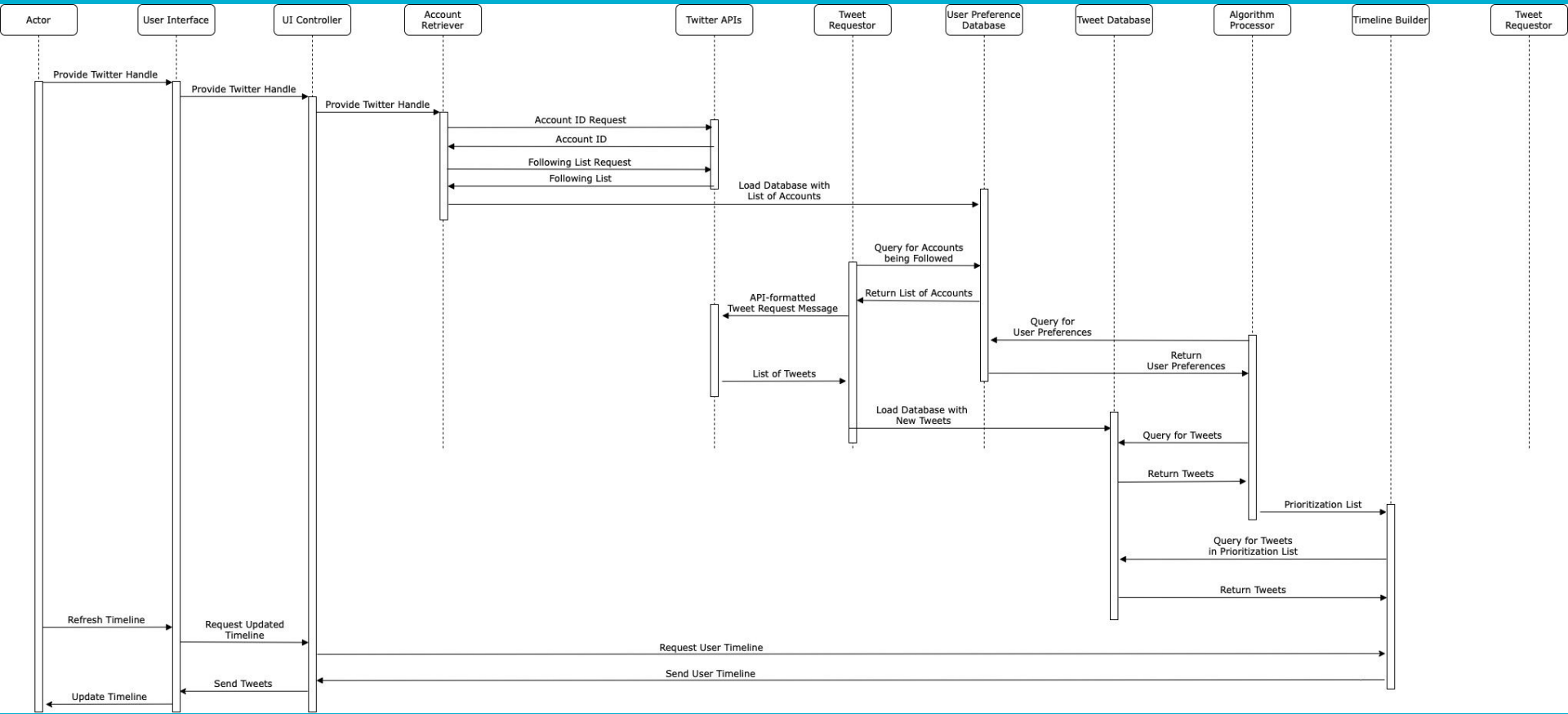
Developed design views based on requirements, mechanisms and risks

- Logical View- (Component diagram)
- Process View - (Sequence diagram)
- Physical View - (Deployment diagram)

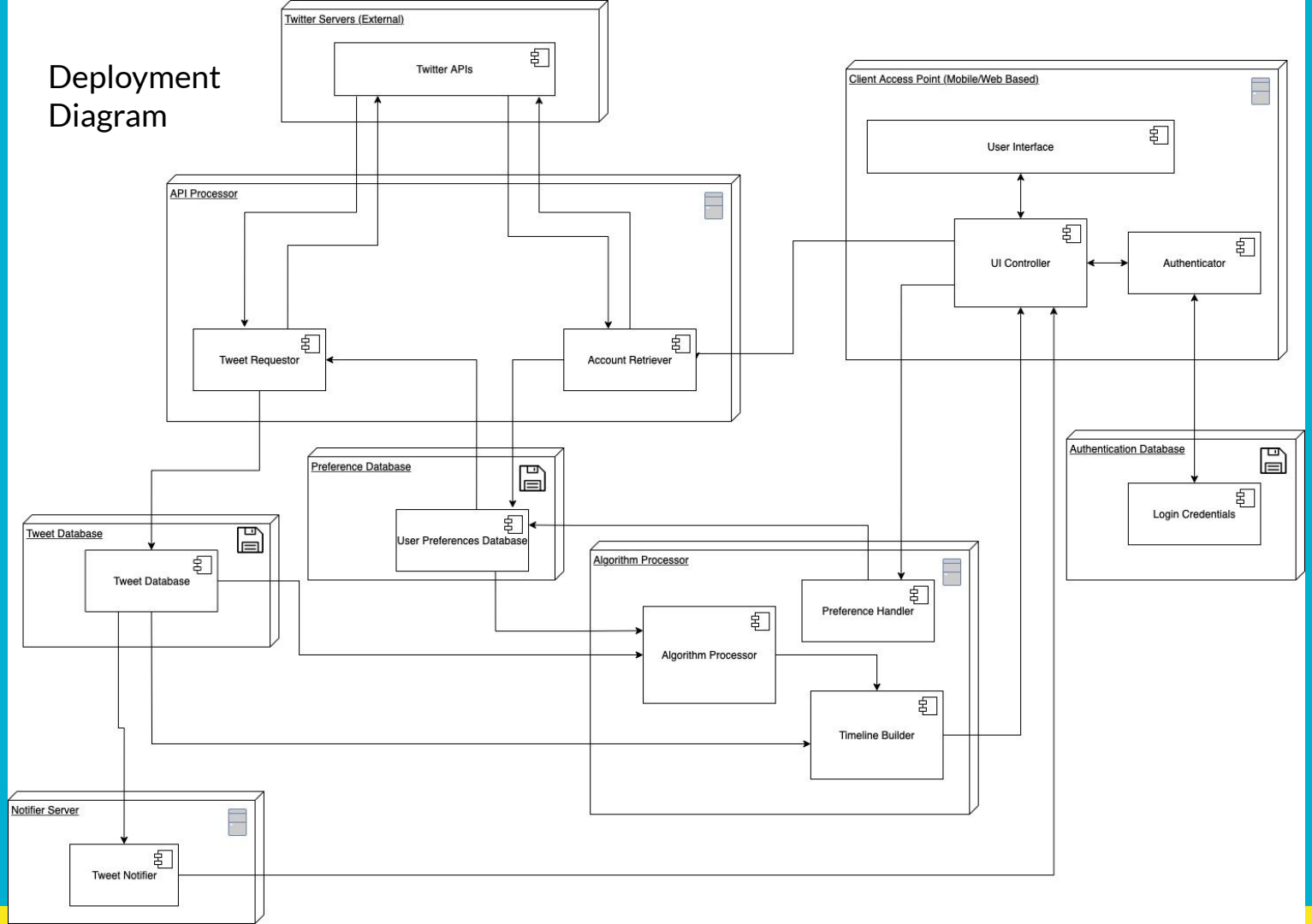


Component Diagram

Sequence Diagram



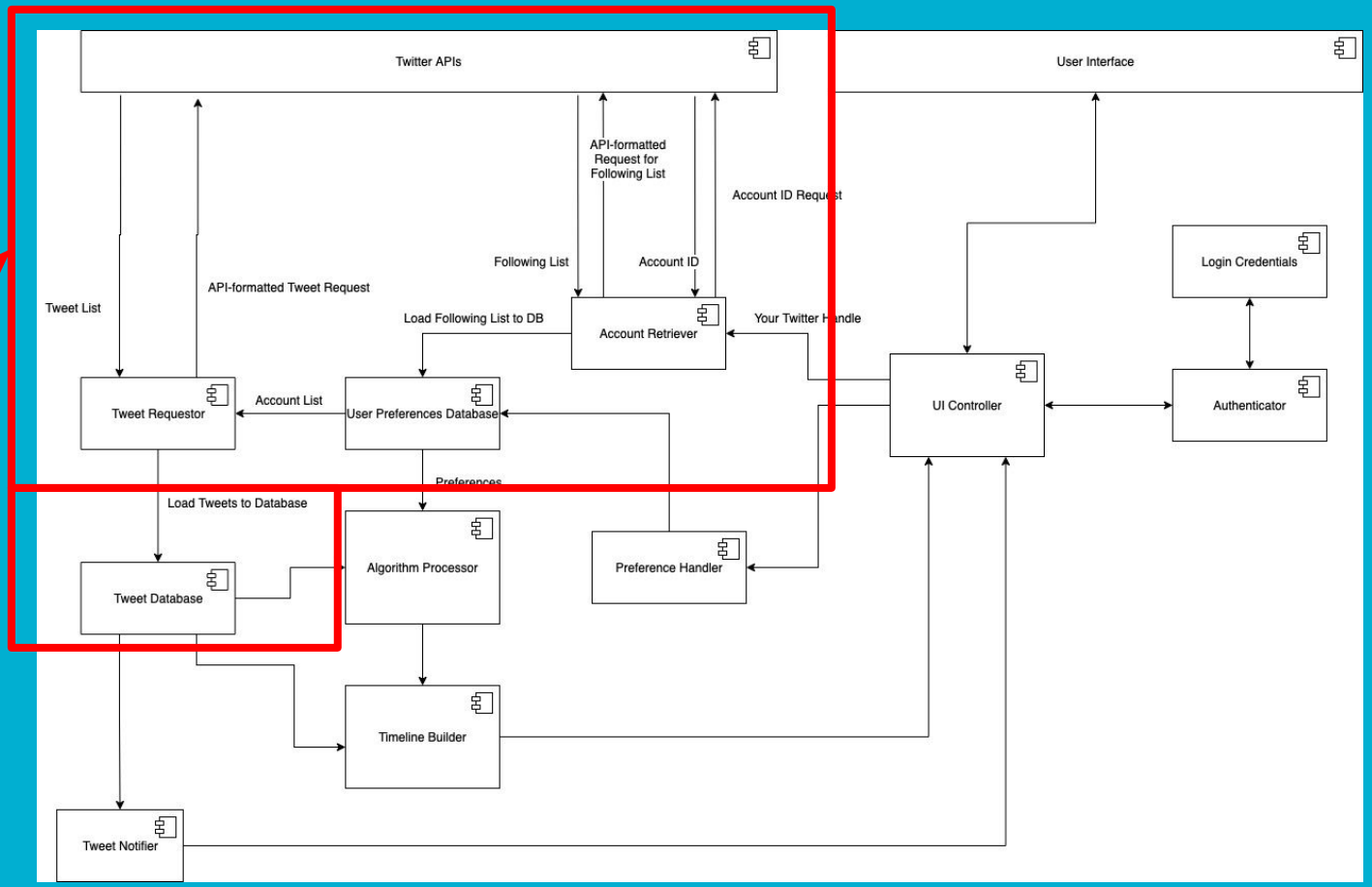
Deployment Diagram



DEVELOPMENT & DEMO

Prototype

Prototype Boundary



Prototype Description

- Programming Languages, Middleware
  - Python3
  - Github
- Obstacles
  - Being able to host a database that we could all share
    - Used CSV files instead
  - Originally thought to use message queue
    - Determined it was easiest to access directly from the components

Demo

Prototype Conclusions

- We frequently hit data access limits during test (300 tweets/15 mins)
  - Architecture would need to account for this better
  - Potential Solutions:
    - Minimizing requests
    - Spacing out times of requests to not reach limits
- A new component could potentially be used to store messages and perform the tasks above

Retrospection

- Having a clear understanding of requirements and having time to time feedback helped
- Aligning of theoretical and practical aspects has assisted in taking better decisions
- Updated design views after seeing alternatives
- NFR’s are equally significant as the FR’s for a better quality software
- Having a good degree coupling and decoupling to manage dependencies and ripple effects within component of the system
- There is no single pattern that fits for every business
- Worked on collaborating tools like - Github, draw io and Lucidchart


THANK YOU