Architectural Requirements Document

Architectural Design Strategy

- Decomposition Strategy

This strategy breaks the system up into smaller factors which makes reasoning large complex problems easier to maintain. This strategy will also allow for ease of reuasable code since the system will be decomposed into smaller parts.

Architectural Strategies

- Multitier Architecture

Multitier allows the system to be broken up into tiers. The three main tiers the application will be broken up into will be the presentation tier, logic tier and data tier. This Architecture allows for technologies to be swapped out easily at different tiers. This is great for future proofing and scaling where is the power app as new technologies arise.

- Model-View-Controller

MVC is mainly used for ease of control when working with GUIs (Frontend related). This will decompose the frontend by having models, views and controllers. This decomposition allows for views and controllers to be reusable and easily testable.

- Service-Orientated Architecture

Since following the Decomposition Architectural Design strategy, the services orientated architecture works best at decomposing the system into different services. This will allow the system to be broken up into different services that can do certain tasks. In the context of Where is the power, scraping loadshedding data from different parts of South Africa is difficult as each munacapility has unique ways of presenting data at different times. This means that having different loadshedding scrapers work in parallel will keep our system running efficiently. This architecure also allows for seperation of concerns regarding business logic.

Architectural Quality Requirements

- Mobility

The application should work anywhere and at anytime with a connected network. The app should be able to work on a desktop and/or a mobile environment.

- Usability

The app should be intuitive enough so that the user can easily use it with out the need of a tutorial. The task of selecting a place to navigate to should be easy for the user to do in a minimal number of steps. What does green overlay mean (no loadshedding), Design choices will determine the usability factor.

- Reusability

Code should be written in a modular manner, so that code can be reused later in the architecture if need by. This will remove the need for redundancy.

- Efficiency

The system should be able to load data without costing to much Processing time on the user side. Load times for data should be optimized by sending small packets of data at time.

- Availabilty

An uptime of atleast 90% is expected of the system.

- Credibility

The user should be able to rely on the application to give accurate routes and accurate information on load shedding schedules. The system should check the loadshedding times from verified sources frequently (Every 30 min)

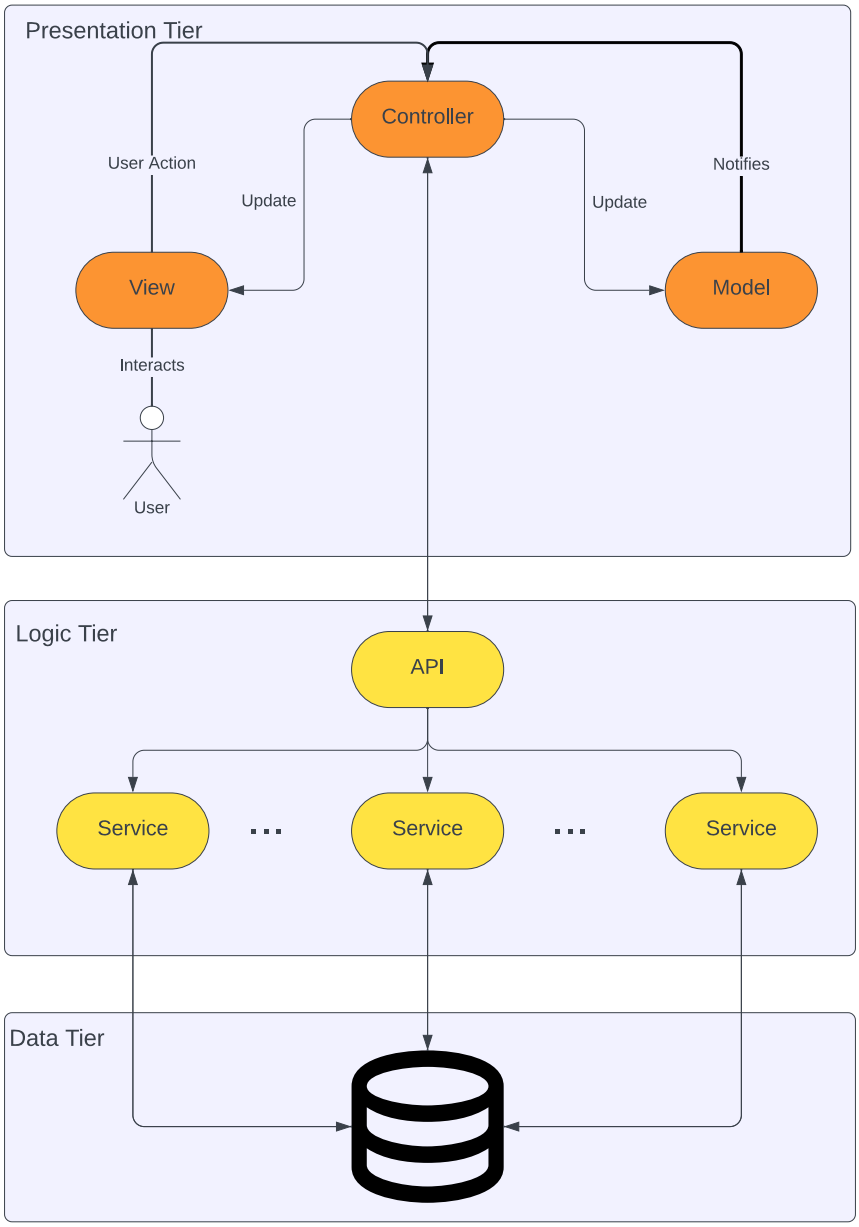
- Security

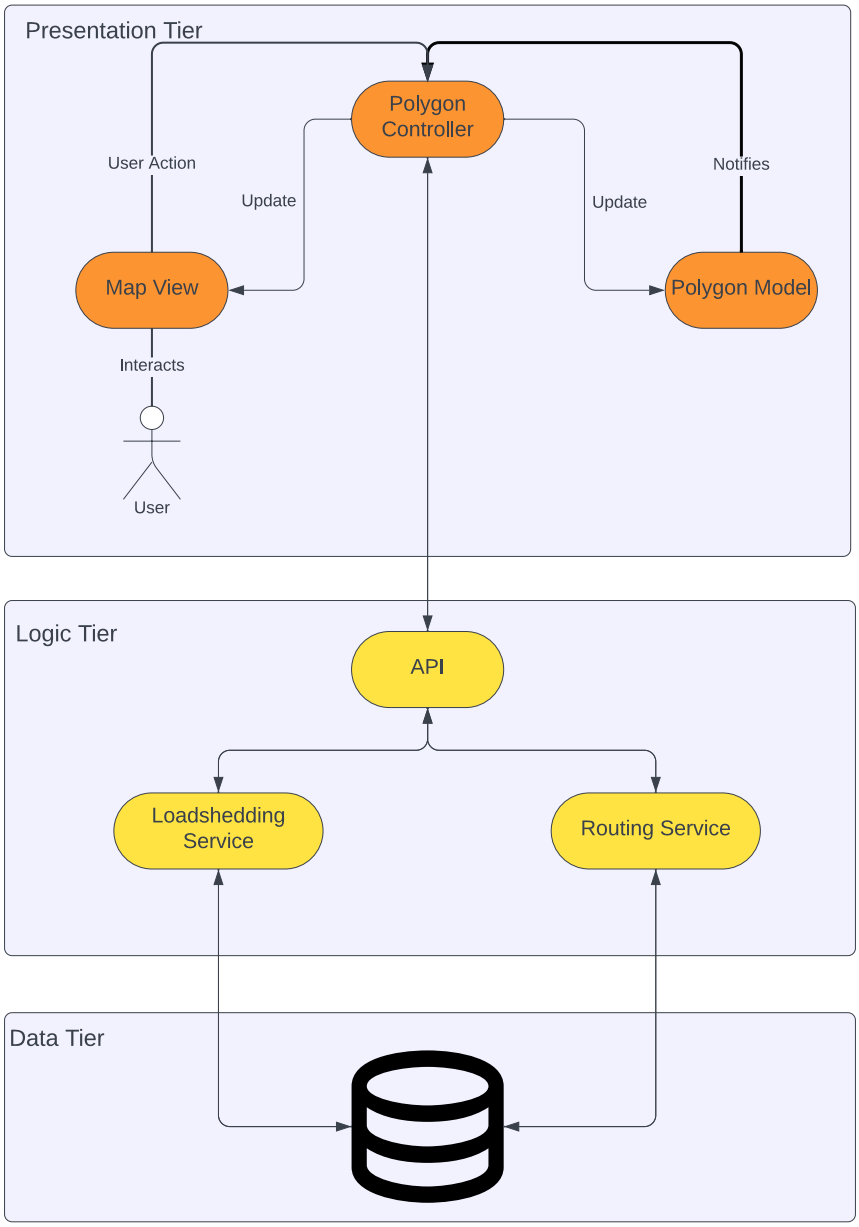
The system should not be exposing data so easily. Sensitive data should be encrypted (Registered users)

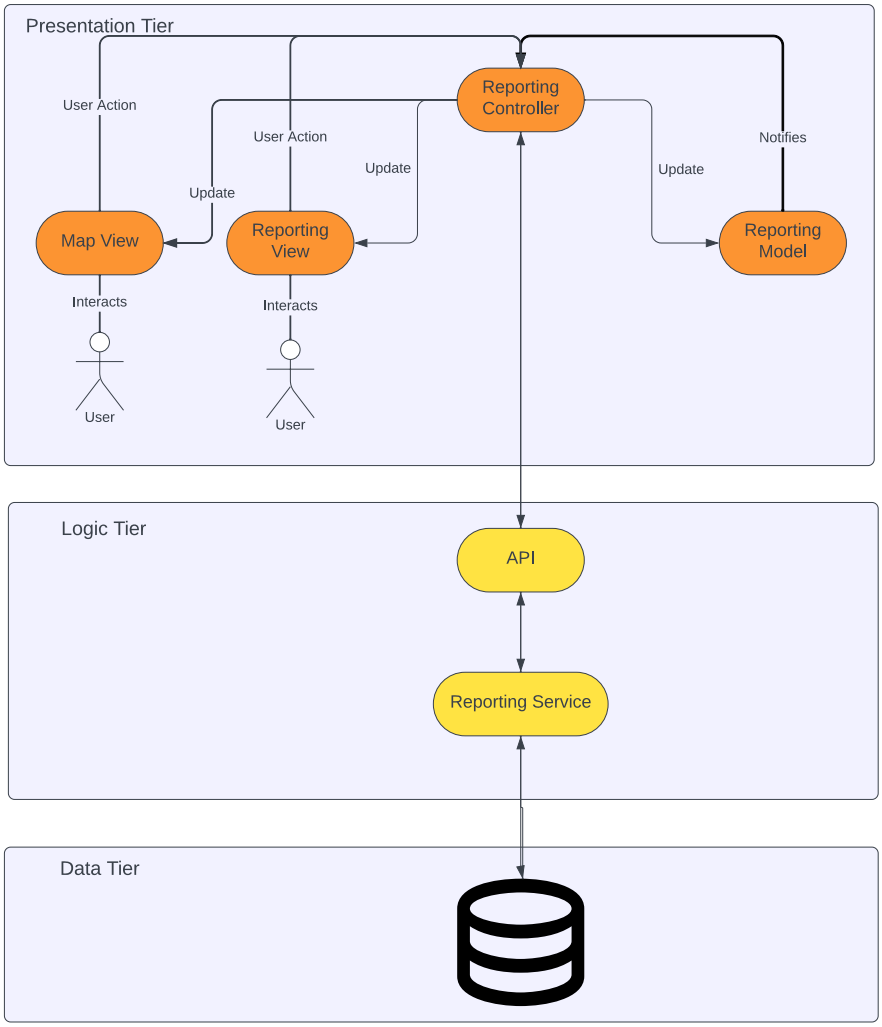
- Scalability

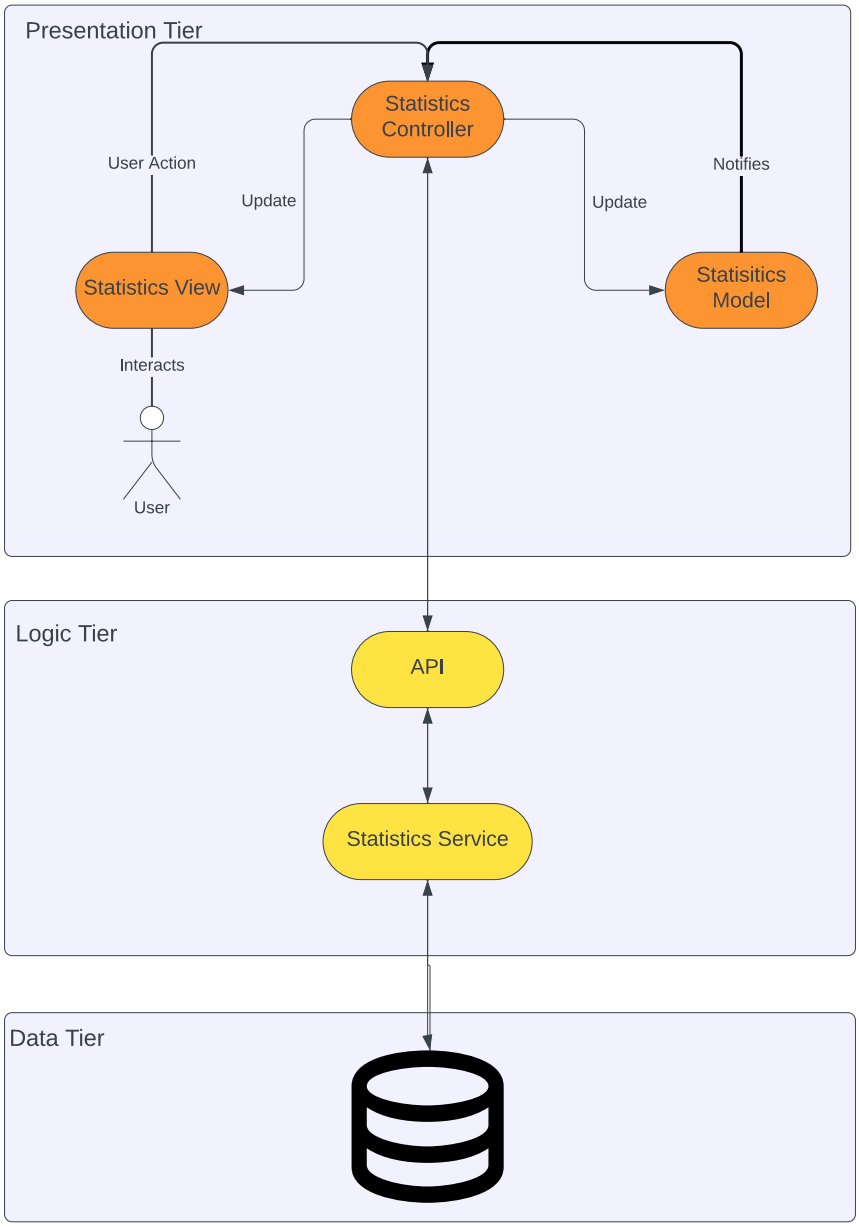
The system should easily be able to pick up more areas (easily) as more web scrapers get loadshedding data from different municipalities. Data should be normalised so that there is consistency when scaling up.

Architectural Design and Pattern


Presentation Tier
Controller
User Action
Update
Update
Notifies
View
Model
Interacts
User
Logic Tier
API
Service
Service
Service
Data Tier


Presentation Tier
Polygon Controller
User Action
Update
Update
Notifies
Map View
Polygon Model
Interacts
User
Logic Tier
API
Loadshedding Service
Routing Service
Data Tier


Presentation Tier
Reporting Controller
User Action
User Action
Update
Update
Update
Notifies
Map View
Reporting View
Reporting Model
Interacts
Interacts
User
User
Logic Tier
API
Reporting Service
Data Tier



Architectural Constraints

- Limit locations to the city of Tshwane

Due to the manual labour and domain knowledge required for spatial mapping, the system will cater for a sample size of South Africa. Time is also a factor as the durationg of the capstone project is limited.

- Mapping data to spatial data

Not all the data we have will be represented becuase of the lack of spatial data. This leads to following point: suburb extensions would be excluded.

- Exclude suburb extensions.

Example: instead of showing the load shedding schedules for Menlopark extension 3, we would only show Menlopark as a whole.

- Out of date Data

All spatial map data used in the system uses a census that was conducted in 2011 (12 years dated).

Technology Choices

Figma – To design wireframes and mockups for planning purposes.

Angular Ionic – Develop Front-End for mobile and desktop web apps

Rust Rocket – API, Reason for using Rust Rocket is guaranteed type safety and speed.

MongoDB – Database to store everything needed.

AWS (Amazon Web Services) EC2 Hosting – Will do all our hosting needs on a virtual Linux machine.

GitHub – Used for version controlling the repository.

GitHub Project Board – For project management, issue tracking and user stories.