

# Night's Watch

## Software Design

**CSCI-P465/565 (Software Engineering I)**

### **Project Team**

**Shantanu Kotambkar**

**Rahul Velayutham**

**Varun Machingal**

**Adarsh Bhandary**

---

## **1. Introduction**

Night's Watch follows agile model for Software development. The project boosts of using JIRA and GitHub repositories for project management, along with PostgreSQL and Django for development.

### **1.1 System Description**

The aspect of Nights watch is to develop a public facing web portal that receives and displays data from the sensors. The sensor data would be overlaid on a map with other data such as weather information. The user would use the portal to display different representations of the data – heat map, accelerated time rendering of the data etc. Additionally, the web system would need the administrative capability to add new sensors. The user in the field would likely use a phone to register the new sensor to a location.

## 1.2 Design Evolution

### 1.2.1 Design Issues

The design issues faced in sprint 1 were:

- Duo Authentication:
  - Limitation: Duo needs you to pay after first 10 users have availed the service.
  - Since, Night's watch is a community project it requires minimal investment, thus Duo authentication is restricted to only first 10 users.
- Database:
  - Implementation of PostgreSQL database turned out to be an issue, as Django form uses SQLite database as default

#### Sprint – 2

- Sensor Images:
  - Limitation: The images need to be in 100\*100 dimension. No other sizes are allowed. Also, there are no available pictures of the spots.
- URL change:
  - URL does not change on navigating to homepage
- Validating invalid token number
- Navigation Bar position:
  - Navigation bar helps people navigate through the website. So, the design of the navigation bar was an issue to consider.

#### Sprint – 3

- Sensor management
  - Fetching the data from API according to the required structure
  - Managing users who can add or remove sensors and based on which parameters can sensors be added
- Weather widget
  - API allows only 60 requests a day
  - Weather data obtained is over a 3-hour period over 3-4 days max
  - More powerful features exist but need to be paid for.
  - Could not provide any security mechanisms for weather.txt
- User management
  - Admin can upgrade other regular users to admin. But giving all admins the right to remove other admins seems like a difficult approach.

#### Sprint – 4

- Coordination with Hardware team
  - The team is trying to coordinate and fetch values from the API which the hardware team is using to provide the required values to be displayed on the website.
  - Currently a vague understanding has been reached but the team is approaching a better solution
- Token Validation bug

- Previously the user could bypass the token submission page to login directly after only one layer of authentication.

### **Sprint – 5**

- Development and testing is limited to one sensor only
- Difficult to accommodate changes to graphs.
- Can E-mail CSV files only to registered email IDs. Django does not allow emails to be sent to unregistered email IDs
- Creating a new thread structure for chat was difficult to understand and implement in terms of design.

## **1.2.2 Candidate Design Solutions**

### **Sprint - 1**

- Authentication:
  - We will be having simple security mechanism for now, but design the project so that dual authentication can be used/implemented in future.
  - The options of Google's two factor authentication or one-time pin are also being explored.
- Database:
  - SQLite is being used in initial stages, usability of PostgreSQL is being explored

### **Sprint – 2**

- User profile:
  - File handling is to be done to manage user profile pictures
- File Handling:
  - We used encrypted= multipart/form-data or else the design issue was that the request files were returning none instead of https response object

### **Sprint – 3**

- Sensor Management
  - Dummy data was used for the time being with parameters as described in the customer meeting along with the sensor development team.
- Weather Widget
  - Decided that since Google Weather API was not feasible to work with, weathermine API was selected to display weather information
- User Management
  - Decided that Admin can add another admin or remove another admin.

### **Sprint – 4**

- Coordination with Hardware team
  - Previously the user could bypass the token submission page to login directly after only one layer of authentication.

- Now the situation is resolved using a decorator.

### **Sprint – 5**

- The API used requires data in a particular order to accommodate new requests, so new functions will have to be created.
- Once more sensors are installed the functionality will be better utilized.
- Unregistered E-mail IDs will not be a situation since all IDs will have a registered E-mail before logging in.

## **1.2.3 Design Solution Rationale**

### **Sprint - 1**

- Authentication:
  - The issue of only first 10 free users being supported by duo authentication system posed a problem for Night's watch, as it would significantly increase the cost of the project. The customer meeting led to the conclusion that the data used in the project doesn't have a vulnerability of theft of sensitive data, as it is an openly available data.
- Database:
  - SQLite works great with Django forms, PostgreSQL needs to be explored more for its usability.

### **Sprint – 2**

- Token based Authentication
  - The problems of validating tokens and denying invalid tokens were looked upon and it was decided that we would be using our own Token based authentication in Django using the mailing server used for password recovery.
- User Profile
  - User profile page was made separately for filling user information.

### **Sprint – 3**

- Sensor Management
  - Data will later be given by the sensor team and the database can be updated accordingly
- We are hosting website on SILO permanently and using SQLite DB

### **Sprint – 4**

- Agreed that a message board is an important and useful feature to have and implemented an app close to a message board
- For weather app, darksky weather was complicated to use so the team used weather data from Weathermine API

## **Sprint – 5**

- The script for fetching data from the API will be able to fetch data from multiple sensors as long as the order in which data is received remains undisturbed
- Graph changes if required can be accommodated.
- Creating instance of topic should now open a new chat with separate chat history.

## **1.3 Design Approach**

### **1.3.1 Methods**

- The Authentication uses Google's salt encryption to encrypt the passwords in the database.
- Django forms are used to build the design framework.

### **1.3.2 Standards**

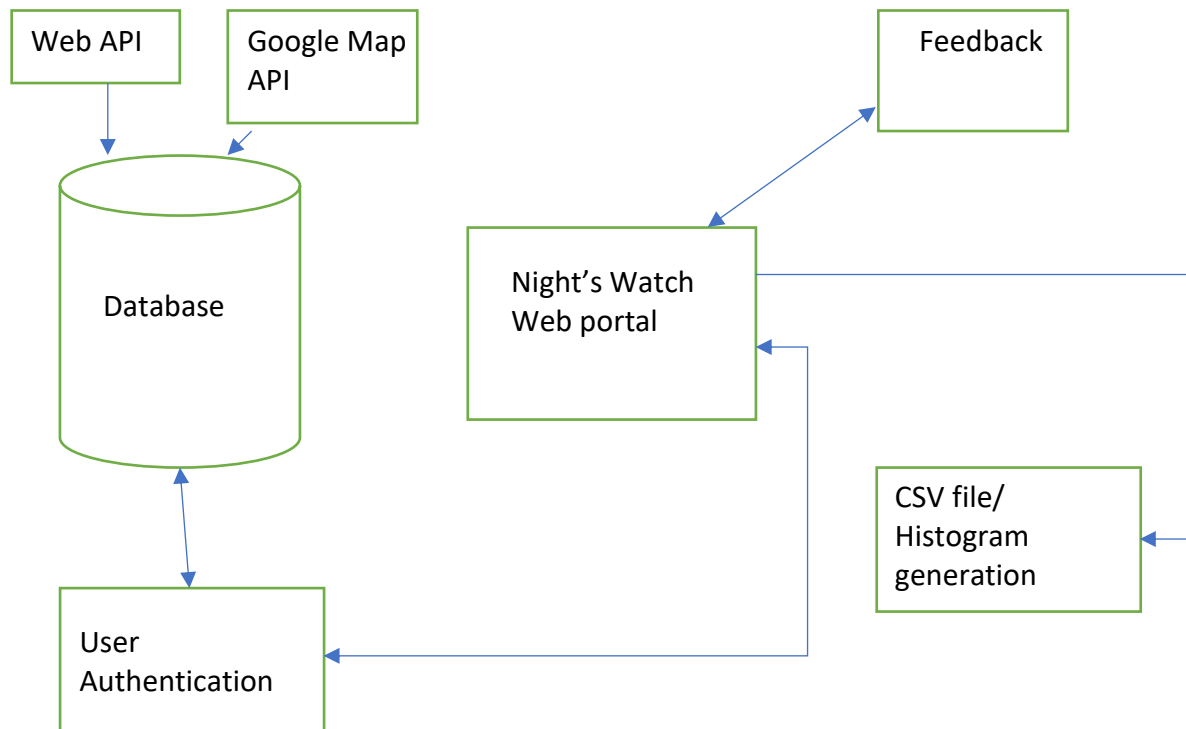
- Python style of coding is used, the design standards specified by EditorConfig.org are adhered. Pycharm IDE adheres to EditorConfig standards.

### **1.3.3 Tools**

- Pycharm IDE is being used for development.
- 
- SQLite for Database development
- StarUML was used for System Architecture.

## 2. System Architecture

### 2.1 System Design



### System Architecture

#### Description:

- Night's Watch is a public facing web portal that receives and displays data from the sensors. The sensor data would be overlaid on a map with other data such as weather information. The user would use the portal to display different representations of the data – heat map, accelerated time rendering of the data.
- The web portal will have authentication feature, and have Admin and User profiles.
- The Admin will be having more functionalities such as changing user's status to administrator, adding new sensors, and managing user profiles.

- Night's watch uses the data provided by the sensors on hourly basis, this data will be sent to the system using web API. The data will then be retrieved and rendered on the map.
- The web portal will provide past record in histograms and csv format files, for further use of data gathered by the sensors.
- Feedback system is implemented to improve the System, and increase the accuracy of the sensor data.

## 2.2 External Interfaces

- Night's Watch receives its data from sensors. This data will be available to our project using a web API, which will be hosting the sensor data on hourly basis.
- The aspects related to the protocol to be used, data structures or timing/handshaking patterns are yet to be explored.

## 3. Component Design

The layout of this section for each component is as follows:

### 1. Component Name: Login/Signup

- **Component Description**

The Login/Signup function is used to register the users and give them access to the system. The first component in our system design is Login/ Signup component. The component uses Django's UserCreationForm() method, to render the registration component.

The registered users have their data stored, and their passwords are encrypted using Google's salt encryption and then stored in the SQLite database.

- **Responsible Development Team Member**

Rahul Velayutham and Varun Machingal are responsible for development of this component.

- **Component Diagram**

Graphically depict the design of the component in terms of interfaces with other components and external interfaces. Also, consider including a diagram depicting the internal operations and/or class relationships in the component.

- **Component User Interface**

If this component includes a user interface, include some details about the interface including what aspects of the component will be available through the interface, a description of each of the user screens that are expected for this component, and a description of each of the user notifications and/or messages that are planned for this component.

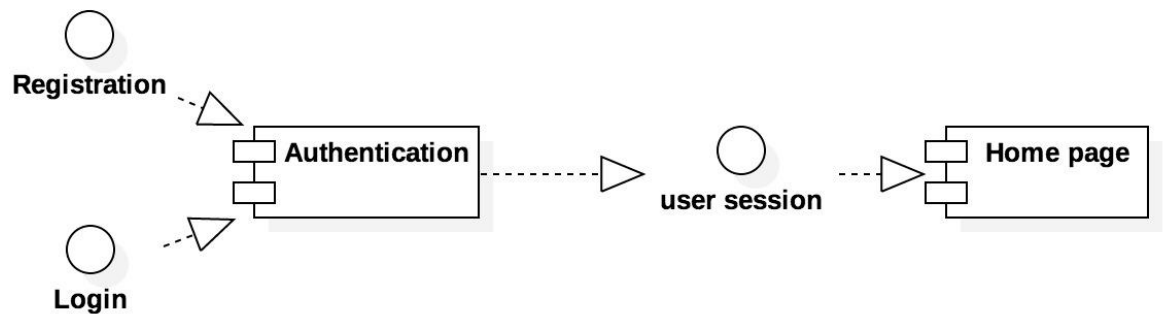
- **Component Objects**

Login/Signup comprises of the classes: register() , home(), profile().

The class register() calls the UserCreationForm() in Django forms, to render registration page to the user. It uses POST method to pass the data and form.is\_valid() function is used to validate the entries.

The class home() is called after user has successfully registered and logged in the system. It uses redirect functionality of Django forms.

The class profile() calls is used to retrieve user data, passed on by the user while registering.



Login/Signup Component Diagram

## 2. **Component Name:** Display.

### **Component Description:**

The Display component will be used to render the sensor data on the maps

### **Component interfaces(internal and external):**

The Display function uses Google maps, on which it renders the data collected from the sensors.

External interfaces: The data is obtained through a web API, thus data has to be extracted from web API into the database.



**Responsible Development Team Member**

Adarsh Bhandary and Varun Machingal are responsible for development of this component.

**3. Component Name:** Feedback**Component Description:**

The Feedback component is another functionality provided to the user to post their feedback regarding the sensor data and their overall experience to help enhance the accuracy of the data, and attain future needs.

The feedback function will use POST method to send the feedback of the user to the admins. The admins will have access to the feedbacks.

**Responsible Development Team Member**

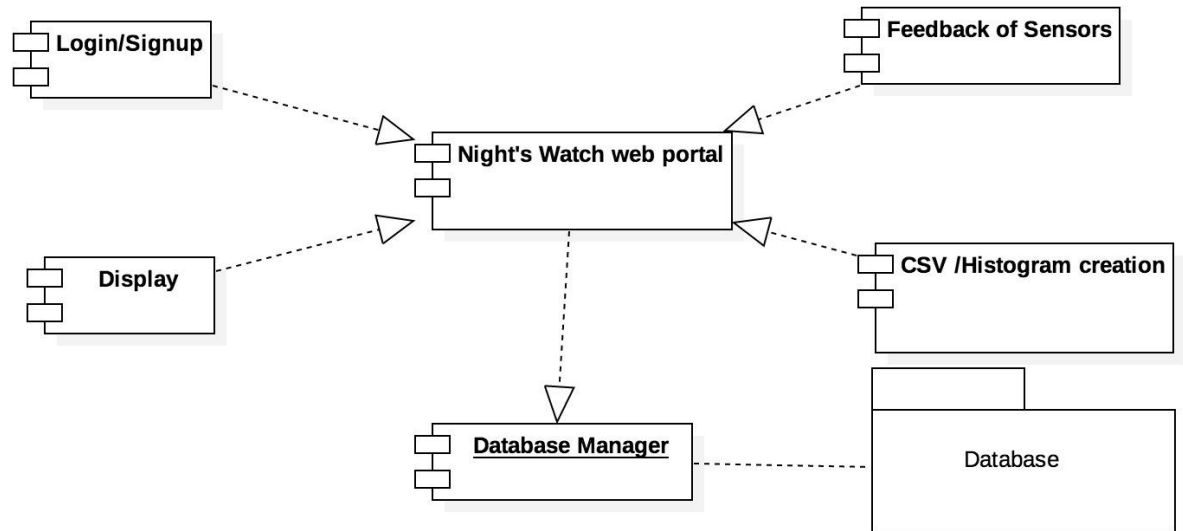
Rahul Velayutham and Shantanu Kotambkar are responsible for development of this component.

**4. Component Name:** Generation of CSV file and Histograms.**Component Description:**

The data received from the web API, will be stored in the database. This history of data, will then be used to generate Histograms and CSV format files of each sensor. This files and histogram can then be used to analysis the data at that sensor location and may be helpful in predicting future sensor locations.

**Responsible Development Team Member**

Adarsh Bhandary and Shantanu Kotambkar are responsible for development of this component.



Component Diagram

## Revision History

Revision	Date	Change Description
Sprint 1	10/01/2017	Component : Login/Signup and sprint 1 module design.
Sprint 2	10/15/2017	Component: Map features, Sensors and User profile
Sprint 3	10/29/2017	Component: Sensor management, Weather widget and User management
Sprint 4	11/12/2017	Component: Weather widget CSS, Message board and Connecting with API for data.