Question 1

We chose Jakob Nielsen’s heuristics because they are the most-used usability heuristics for user interface design, broad rules of thumb and not specific usability guidelines.

1. **Appropriate set of heuristics:**
    1. Visibility of system status – user being informed about what is going on through appropriate feedback, such as real time data on display.
    2. Match between system and the real world - system should speak the user’s common language, with words, phrases and concepts familiar to the user.
    3. User control and freedom - "emergency exit" to leave the unwanted state without having to go through an extended dialogue, such as back button.
    4. Consistency and standards - users should not have to wonder whether different words, situations, or actions mean the same thing or where to find certain things such the menu bar or logout button.
    5. Error prevention - eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
    6. Recognition rather than recall - minimize the user's memory load by making objects, actions, and options visible such as using a light icon instead of LDR.
    7. Aesthetic and minimalist design - webpages should not contain information, which is irrelevant or rarely needed, must focus on the main concept.
    8. Help and documentation – Any documentation such as information should be easy to search, focused on the user's task, list concrete steps to be carried out and what the system is about.

2. Evaluation based on our prototype:
    1. Match between system and the real world - system speaks the users' language, with words, phrases and concepts familiar to the user such as using light sensor instead of LDR.
    2. User control and freedom – system does have an emergency exit option which the back button/option in a webpage linked to the homepage.
    3. Consistency and standards – system is not consistent in terms of positioning functions in familiar positioning and naming of common items such as alerts instead of notifications.
    4. Error prevention – no error prevention messages.
    5. Recognition rather than recall – there is recognition in terms of icons used, easy to remember the information on the webpage.
    6. Flexibility and efficiency of use – the easy understandability of the system, I’d say it’s easy to understand. But other than that, the system does accommodate both user experiences.

7. Aesthetic and minimalist design – our system is very minimal, use of icons and less texts.
8. Help users recognize, diagnose, and recover from errors – our system doesn’t have this option.
9. Help and documentation – our system does not need this option because everything is laid out easily and the icons help with understanding how the system works.

Question 2 – Prototype

The new prototype does satisfy user requirements such as having documentation that explains how the system works, user getting a feedback on tasks performed and the Nielsen’s heuristics that are appropriate for the system.
https://xd.adobe.com/view/1160e1db-ad3b-4b9e-5c04-de880cb15439-6751/

Question 3

- In the previous evaluation the user encountered a problem of not knowing how to toggle the lights on and off, not knowing the task to perform. The heuristic evaluation agreed that the user couldn't perform that particular task because there was no documentation to guide the user.
- Users encountered a problem of not having more options with opening the images or the home button leading nowhere.
  The heuristic evaluation yielded the same findings that there are some buttons that lead nowhere when clicked, but it is not explicitly shown that you cannot click these buttons. The experts that did the evaluation decided that instead of the user being frustrated clicking a button that leads nowhere, they should be made aware that a certain option isn't clickable when it isn't.

We prefer the heuristics evaluation because it is more comprehensive than the usability testing approach. In the heuristics evaluation, the experts were able to better assist us in diagnosing the problems with using the system and further advised us here and there on what to do to combat the problems. In the usability evaluation, users did not really nitpick the smaller details of the system.

Question 4

Evaluation approaches		
Summary	***Usability Testing***	***Field Studies***

Findings	• Actionable findings to redesign/improve our system for example one user suggested the use of a big font size.	• Learn the unexpected by observing users in their natural environment
Benefits	• • • Full control of users. Time efficiency Provided unbiased examination of our product.	• It yields very detailed collection • of data. Realistic expectations • Allow system to be tested under realistic conditions.
Costs	• Barely any costs.	• It cost time
Limitations	• Only small sample of potential users can undertake. • Users have different preferences and opinion about the product.	• It is up to the participants to decide how to use the product and when.

Application of Software Engineering:

Principles:

- Abstraction – reduces complexity such as the user having to interact with the webapplication user interface and not see the complexity behind with sensors.
- Consistency – familiarity with the functions, type of icons used and the position of functions such as the logout page.
- Hiding – a secondary user has functions they cannot access for security reasons such as adding another user, only an admin can do so.

Umbrella Activities:

- Project Tracking - we using an online communication tool and we have regular meetings to track the progress of the project and keep up to date.
- Technical Reviews - technical issues that / may arise are discussed in the meetings we conduct, and solutions to those problems/issues are discussed.
- Work Product Preparation and Production – we evaluated the feedback that we received in the previous week and made changes to our product accordingly. For this part of the project we had to rebuild the prototype taking into consideration a heuristic evaluation that we did.

Goals:

- Understandability – the interface is easy to interact with, it makes use of icons with easily readable labels to make it easier for the user to interact with the system. The user doesn't have to try and remember where everything is located or what performs what function because it is clear and easy to see.
- Portability – the system is portable because it can be accessed anywhere since it's web-based.
- Security – The user uses the login details, password and email to access the system.

Process – Agile Process, the team is self-organized, cross-functional and we have a collaborative effort with end-users in a sense that we ask for their input.
We make use of the agile process because in each week, we iterate on the previous week's work instead of delivering software all at once. The work is done in short 1 week cycles, where each week we have the work split up and each person does a certain task and works with someone else in the team based on how their tasks relate to each other. The development is user-centric in the sense that we involve users in each step of the development process to check whether or not the product meets the user requirements. We make an effort as a team to meet each week (and take meeting minutes) and discuss the development process and check in where we are in terms of developing the product.

DATABASE

```
CREATE TABLE user(        userId
INTEGER PRIMARY KEY,  username
TEXT NOT NULL,  password TEXT
NOT NULL,  type INTEGER NOT
NULL,  number TEXT
);

CREATE    TABLE    room(        roomId
INTEGER  PRIMARY  KEY,   name  TEXT
NOT   NULL,     userId   INTEGER   NOT
NULL,
        FOREIGN KEY(userId) REFERENCES user(userId)
);

CREATE TABLE `sensor` (
        `sensorId` INTEGER PRIMARY KEY,
        `name` TEXT NOT NULL,
        `type` INTEGER NOT NULL,
        `description` TEXT NOT NULL,
        `roomId` INTEGER NOT NULL,
        FOREIGN KEY(`roomId`) REFERENCES `room`(`roomId`)
);

CREATE TABLE `event` (
        `eventId` INTEGER PRIMARY KEY,
        `type` INTEGER NOT NULL,
        `sensorId` INTEGER NOT NULL,
        FOREIGN KEY(`sensorId`) REFERENCES `sensor`(`sensorId`)
);
```

Internal Model

```
-- ************************************** `User`

CREATE TABLE `User`
(
 `userID`      int NOT NULL ,
 `username`     varchar(40) NOT NULL ,
 `userPassword` varchar(20) NOT NULL ,
 `userType`     varchar(45) NULL ,

PRIMARY KEY (`userID`)
);



-- ************************************** `Room`

CREATE TABLE `Room`
(
 `roomID`   int NOT NULL,
 `roomName` varchar(45) NOT NULL ,
 `userID`   int NOT NULL ,

PRIMARY KEY (`roomID`),
FOREIGN KEY (`UserID`) REFERENCES `User` (`userID`)
);



-- ************************************** `Sensor`

CREATE TABLE `Sensor`
(
 `sensorID`      int NOT NULL ,
 `sensorType`    varchar(45) NOT NULL ,
 `sensorName`    varchar(45) NOT NULL ,
 `sensorDescrip` varchar(45) NOT NULL ,
 `roomID`        int NOT NULL ,

PRIMARY KEY (`sensorID`),
FOREIGN KEY (`roomID`) REFERENCES `Room` (`roomID`)
);




-- ************************************** `Event`

CREATE TABLE `Event`
(
 `eventID`   int NOT NULL,
 `eventType` varchar(45) NOT NULL DEFAULT 0 ,
 `sensorID`  int NOT NULL ,
PRIMARY KEY (`eventID`),
FOREIGN KEY (`sensorID`) REFERENCES `Sensor` (`sensorID`)
);
```