EE 461L Team Project
Stakeholder Needs and Grading

Version 5.01162023

Motivation:

The well-known Software-as-a-Service (SaaS) model has now expanded to several online services, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Hardware-as-a-Services (HaaS). With the cloud now supporting these and many other services, the term Everything as a Service (XaaS) was coined to refer to the extensive variety of emerging services and applications that people can access on demand over the Internet as opposed to being housed on-premises. According to a new report by Research Insights titled "Global Everything-as-a-service (XaaS) Market Research Report 2019-2026", the XaaS market is expected to grow to more than $345 billion (about $1,100 per person in the US) over the next few years. Many industries are now adopting the on-demand approach of acquiring services through cloud computing because it offers agility and flexibility, allowing companies to acquire technology quickly and with fewer up-front costs than they would with a purchase and license agreement.

Team Project:

In this project, we will bring together the tools and techniques (marked in italics in this document) that we are learning in the class to implement a Proof of Concept (PoC) for a web application (henceforth, referred to as the app) for a functioning HaaS system. This PoC app is inspired by the University of Utah POWDER program (https://powderwireless.net/). The overall nature of the app is predefined, but your team can be creative in adding features that exceed the *stakeholder needs*.

A simplified software architecture of the app is shown in Figure 1 below

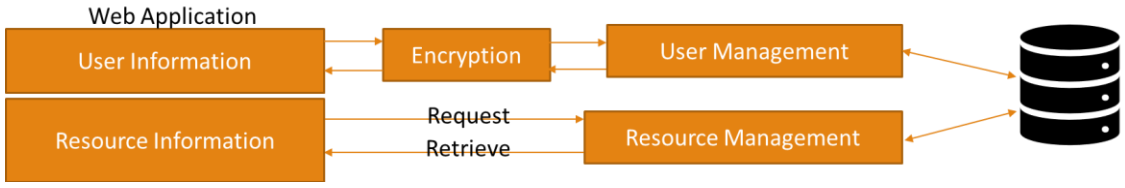

Figure 1: Simplified Software Architecture of a HaaS app

This project will be done with an assigned project team, and you will submit just one deliverable for the entire team during each phase. Each team should contain no more than 5 students. Based on the size of the section, there may be a few teams that have 4 members. All the team members should be in the same section. This will help with coordination as you will have to Identify 2 one-hour blocks when you can all meet weekly

HaaS System Project Description

Create a PoC for an app for a functioning HaaS system to enable users to achieve the following *Stakeholder Needs* (SN).

SN1: Create and maintain secure user accounts and projects on the system
SN2: View the status of all hardware resources in the system
SN3: Request available hardware resources and datasets from published sources
SN4: Once approved, checkout and manage these resources
SN5: Check-in the resources and get status of all hardware resources in the system
SN6: Deliver PoC within *schedule constraints*, with support for scalability

Based on these stakeholder needs; your team will define system requirements. The table below lists some of the system requirements to help you get started. You can use these requirements as a starting point and further refine them and add more. As discussed in the class, you should map each requirement to one or more stakeholder needs. Likewise, you should ensure that all stakeholder needs are met.

System Requirements	Stakeholder Needs Met	Related Course Modules
SR1: PoC shall be delivered within budget and schedule constraint, with periodic updates to stakeholders	SN6	SDLC (Software Development Life Cycle), Agile
SR2: PoC App shall have a front-end web application that allows users to enter inputs and views outputs	All	Web UI Development, React.js
SR3: PoC App shall have a mechanism for encrypting user-id and password	SN1	Python OOP (Object Oriented Programming), Python Modules (Cryptography)
SR4: PoC App shall have a mechanism for creating new projects or accessing existing projects	SN1	Python OOP, Python Modules
SR5: PoC App shall have a database for maintaining user login credentials, project codes, project details, resource details	SN2, SN3, SN4, SN5	Python, MongoDB

Minimum Viable Product

The features described next are needed to deliver a *Minimum Viable Product* (MVP) for the PoC. Note that you can be creative in expanding the scope of these features.

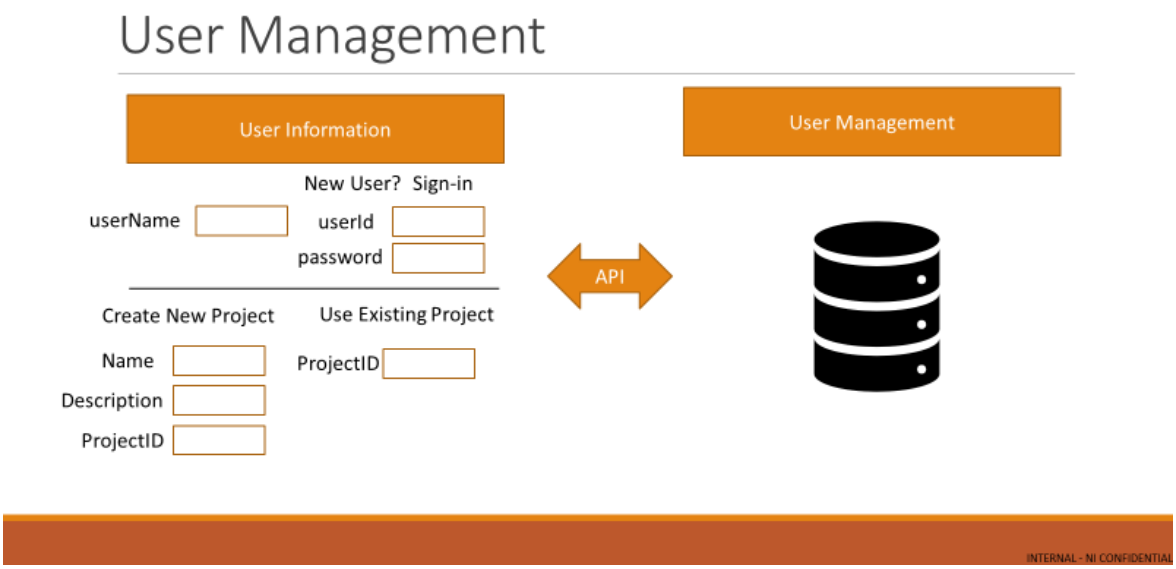

Figure 2: User Management

The user management section, shown in Figure 2, should have the following features:

1. A sign-in area where users can sign in by providing their user name, userid and password. If the user clicks on New User, display a pop-up that allows them to enter a new userid and password.
2. An area where users can create new projects, by providing project name, description, and projectID.
3. An area where users can choose to login to existing projects.
4. A database where you can save user information and project information.
5. An API to access information stored in the database.
6. Security features to encrypt the userid and password.

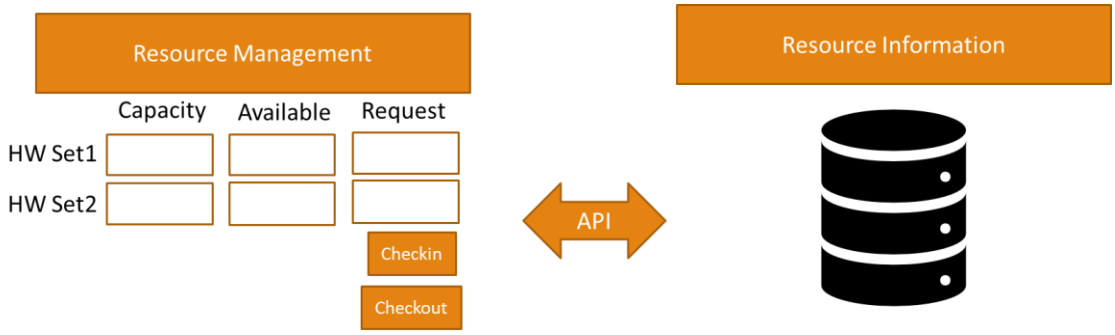

Figure 3: Resource Management

The resource management section, shown in Figure 3, should have the following features:

1. A display area which shows the capacity of HWSet1 and HWSet2.
2. A display area which shows the availability of HWSet1 and HWSet2.
3. A database where the HW information can be stored and can be retrieved from.
4. A display area which shows how many units of HWSet1 and HWSet2 user wants to checkout and later check-in.

General Requirements

- One repository for the project - used for all three phases. Please add us (Dr. Samant and the TAs) to your repo.

- Collaborate with your team using:
    - daily standups, for example using Zoom or Slack
    - GitHub boards
- Use an issue tracker to keep track of open issues. An issue is a shortfall which prevents the software from fulfilling its required function. Use issues to track bugs and needed improvements.
    - Depending on the bug or improvement, an issue may or may not need to be closed in a particular project phase (it depends on the phase requirements.)
    - You may want to use a project board to track issues. Please do not combine them on a board with user stories.
- User stories - on project boards
    - Ideally, a user story is fine-grained so you can more easily estimate the time to completion and assign it to one team member or pair.
    - All the user stories should be defined by the end of Phase1. User stories can be refined as you work on the project. The user stories for each phase correspond to work that will be started and completed during the phase.
    - Read https://www.mountaingoatsoftware.com/agile/user-stories carefully!
        - User story must be described in three sentences or less

- discuss, include assumptions, refine

You are strongly urged to use tools learning during the class, such as Python, React.js, MongoDB, PyTest, Heruko Cloud Deploy. If your team would like to use a different stack, please discuss this with your TA.

Specific Phase Requirements and Grading

Phase 1 (5 pts)

Rubric

Topics	**Points**
R1-1: Project Plan. Should include information such as team members, sprint velocity, collaboration tools used, implementation methodology used and so on	**2**
R1-2: All Features on a board. Initial work items (user stories, technical debt, or research items) created for features.	**2**
R1-3: High level sketch (on paper, PowerPoint, draw.io, Visio) of the application	**1**
Total	**5**

Phase 2 (8 pts)

All general requirements for the project must be satisfied at the end of this phase

Rubric

Topics	**Points**
R2-1 Hardware resources are stored in the database. API created for app to access this information from the database.	**3**
R2-2 User information (account and project) that you have collected and stored in your database is accessible from your app. No hard-coded data on pages.	**3**
R2-3 App is hosted on the cloud and accessible to TAs/instructor via an URL	**2**
Total	**8**

Phase 3 (7 pts)

Rubric

Topics	Points
R3-1 Describe how the app is scalable across all aspects of the project (database, front end UI, back end). Some specific use-cases of scalability can be discussed in the context of 1. How to add more hardware resources. 2. How can the look and feel of the app be modified by the stakeholders (such as background or adding more text). 3. How can the billing information feature be added so that users can see how many credits have been used for the hardware resources	**1**
R3-2 User documentation.	**1**
R3-3 Demonstrate test coverage (unit tests for JavaScript and Python code)	**1**
R3-4 Continuous improvement. Describe how your team used concepts of refactoring to improve the code since completion of Phase2	**1**
R3-5 Peer and Self-Review 1. 2 point for peer review (average of peer review scores) 2. 1 point for self-review (scaling function of delta between the score you gave yourself and the average of your team's score for you)	**3**
Total	**7**

Useful Design Pattern Considerations:

Apply the principle of *information hiding* in your project design. Some things you may want to consider and describe during Checkpoint3 are:

- Which design decisions do/did you anticipate may change, and how have you encapsulated these aspects within modules?
- Does your modularization scheme make your program robust to changes so that additional features can be easily added? How?
- What disadvantages does your modularization approach have? How will you mitigate them?
- How would you incorporate Refactoring in your code? During Checkpoint3 discussions, you should highlight a few example candidates for refactoring.
- Cite your sources: What sources did you use? Provide links to tutorials, books, Stack Overflow pages that you used and indicate how each source was used.

- Reflection: What did your team learn? Three things you want to continue doing, three things you struggled with and need to improve?
- Your team will present your application during the last week of class. You must be present and participate in a significant way to receive credit. Checkpoint3 discussions are limited to 30 minutes.