

Team Project

Stakeholder Needs and Grading

Version 20250903

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-Service (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.

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

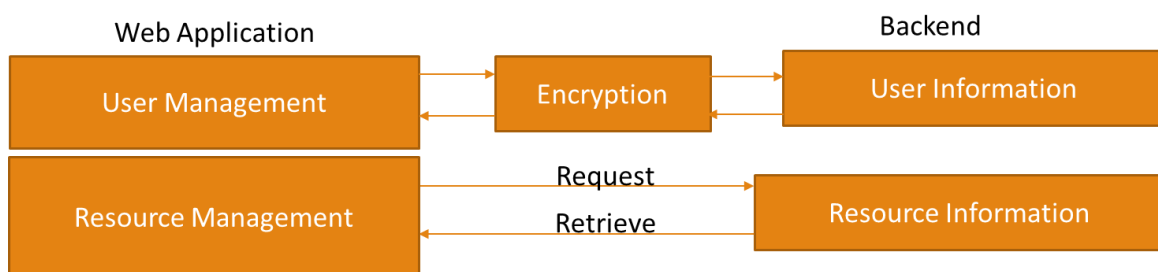


Figure 1: Simplified Software Architecture of a HaaS app

This project will be done with your chosen project team, and you will submit just one deliverable for the entire team during each phase. Each team should contain no more than 5 to 6 students.

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).

SN0: Accepted *quality and reliability metrics*

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

SN4: Checkout and manage these resources

SN5: Check-in the resources and get status of all hardware resources in the system

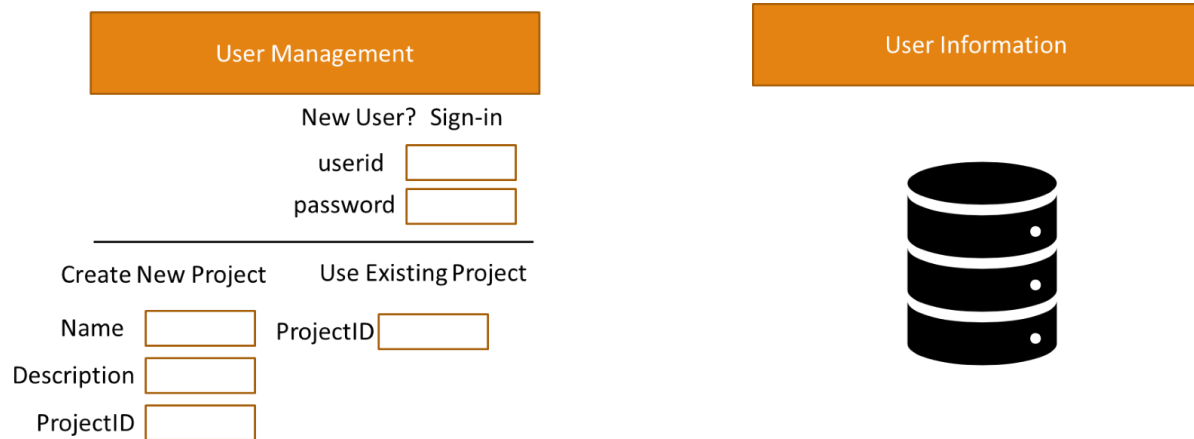
SN6: Deliver PoC within *schedule constraints*, with support for scalability

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	Notes
SR1: PoC shall be delivered within budget and schedule constraint, with periodic updates to stakeholders	SN6	
SR2: PoC App shall have a front-end web application that allows users to enter inputs and views outputs	All	
SR3: PoC App shall have a mechanism for encrypting user-id and password	SN1	
SR4: PoC App shall have a mechanism for creating new projects or accessing existing projects	SN1	
SR5: PoC App shall have a database for maintaining user login credentials, project codes, project details, resource details	SN2, SN3, SN4, SN5	

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.



The diagram shows a 'User Management' section with a sign-in form and a 'Project' creation form, and a 'User Information' section with a database icon.

User Management

New User? Sign-in

userid

password

Create New Project Use Existing Project

Name ProjectID

Description

ProjectID

User Information

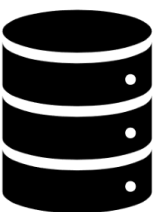
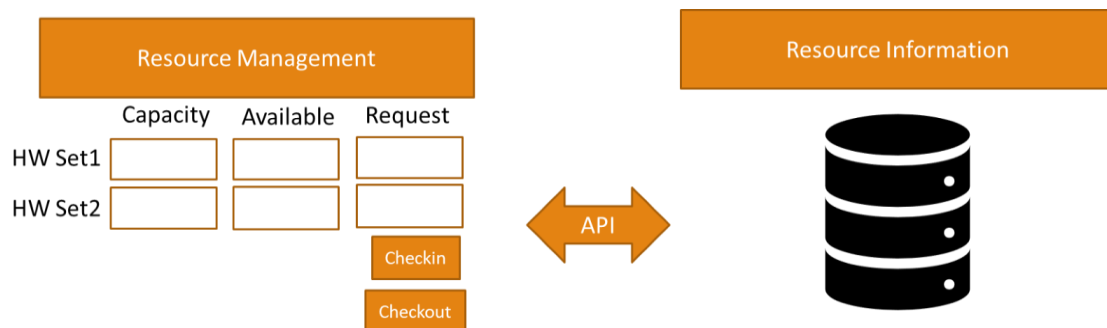


Figure 2: User Management

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 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.

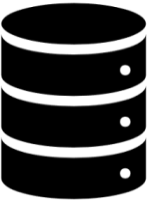


The diagram shows a 'Resource Management' section with a table and buttons, and a 'Resource Information' section with a database icon.

Resource Management

	Capacity	Available	Request
HW Set1	<input type="text"/>	<input type="text"/>	<input type="text"/>
HW Set2	<input type="text"/>	<input type="text"/>	<input type="text"/>

Resource Information






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.
5. HW capacity is a global quantity shared across all user projects

General Requirements

- One repository for project - used for all three phases. Please add us (Dr. Samant and the TAs) to your repo.
- Collaborate with your team using tools such as :
 - 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 Points)

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	1
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
R1-4: Choice of tools and approach	1
Total	5

Phase 2 (15 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.	5
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.	5
R2-3 App is hosted on the cloud and accessible to TAs/instructor via an URL	5
Total	15

Midterm2 will contain some questions related to your understanding of the Team Project implementation. Some things you may want to think about in your preparation for midterm2 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? In your project presentation, you should highlight a few example candidates for refactoring.
- Reflection: What did your team learn? Three things you want to continue doing, three things you struggled with and need to improve?