Requirements Analysis Document (RAD)

ELEC 376 (week 2)

About the project

- You now have your group and TA assigned
- You started thinking about your project idea
- Reach out to your TA for help with the scope if needed

Important dates

- Week 2: Project definition: due on Thursday, Sep 11th
- Week 3: Requirement Analysis Document due on Saturday, Sep 20th
- Week 3: Sprint 1 officially starts

About the project description [Due on Thursday]

- **Project Overview:**
  - **Project Vision**
    - What is the high-level goal or purpose of the project?
    - Why is this project important or valuable?

About the project description [Due on Thursday]

- **Scope Definition**
  - What features or functionalities will be included in the project?
  - What features are explicitly out of scope (not included)
- **Target Audience**
  - Who are the intended users of the software?
  - What are their primary needs and expectations?

About the project description [Due on Thursday]

- **Communication**
  - Did you discuss your project idea with your TA/mentor? [Yes/No, provide brief details]
  - What means of communication will you be using within the team? (e.g., Slack, teams, email, weekly meetings)

Overview

- Getting started on your project:
  - How to come up with **functional** and **non-functional** requirements?
  - How to document these requirements?

Before you start

- You should agree on a project with your team
- Normally Requirements are obtained from your client - but for this course, you are your client!
- So, pretend: If you were buying this system, what would you expect it to do?

Requirements

- Functional Requirements

- Non-Functional Requirements

Requirements

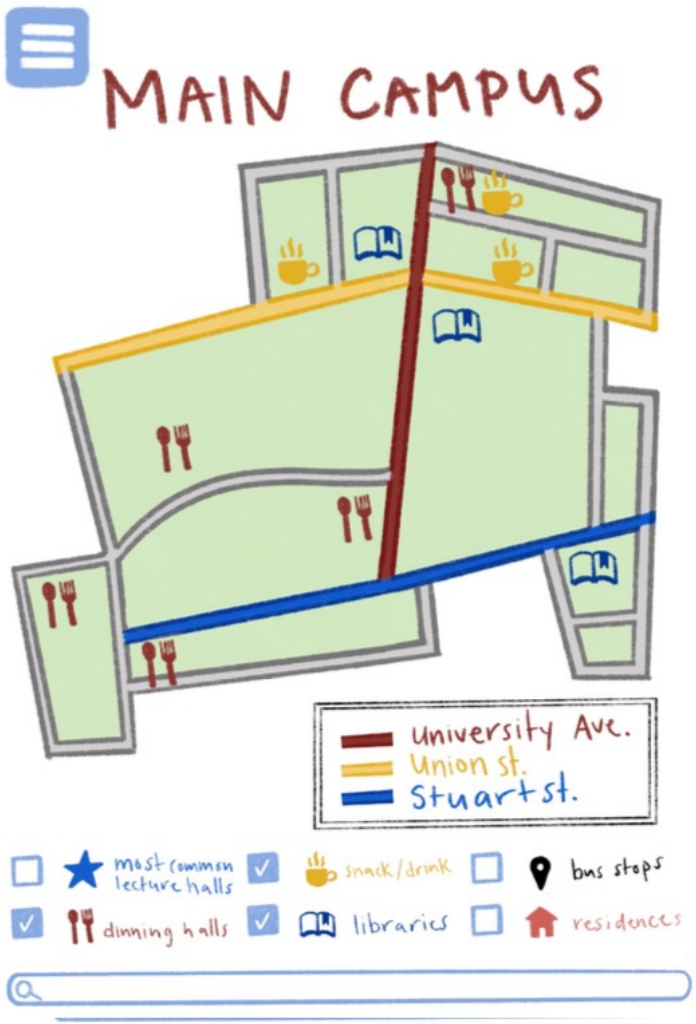
- Functional Requirements: **What** your system does?
  - Goals, features
- Non-Functional Requirements: **How well** your system does what it does?
  - Boundaries around your functional requirements
    - Processing time, memory usage, # concurrent users, maintenance, UX

Requirements, Cont.

- So, as you are listing functional requirements, you must always **consider the restrictions** on those requirements.
- It is important to separate requirements along these boundaries because the requirements will be treated differently:
  - Functional Requirements result in tasks that must be completed. Testing will verify that the task is complete.
  - Non-Functional Requirements result in tests directly. Tasks will be created to address the problem of a non-functional requirement that is not met.

Requirements, Cont.

- How to come up with requirements?
- Try to think through every aspect of your system operation.
- Sketch out GUI prototypes.



Requirements, Cont.

- Flowcharts to describe how the system operates.

Requirements, Cont.

- Every requirement must be **verifiable**. Can you create a test that verifies that the requirement has been met?
- Everyone has to agree on what a requirement means - this often is a challenge for the system architect to make sure he has clearly communicated his vision of the system.
- **Communicate!!!!**

Requirements, Cont.

- Don’t worry yet about how long a requirement will take to implement or who is going to do it.
- Requirements are not just about writing software, you may require other tasks **such as research, algorithm choice and testing, art, writing a backstory, creating a help document**, etc.
- Later, you will consider the sequence of tasks, as well.

Documenting requirements

- Everyone on the team needs to see and agree on the requirements.
- We will use two techniques to list these requirements:
- The first is the RAD - **“Requirements Analysis Document”.**
- It is a bit old-fashioned, but it is easy enough to create and will force you to think through your system.
- It is graded!

Documenting requirements

- The second technique is more modern and more likely to be what an Agile team would do:
  - You can create issues on GitLab within milestone (i.e., sprint) that you commit to implementing

Project deliverable – the RAD

- "Requirements Analysis Document".
- The purpose of this deliverable is to make sure you have started on the design of your application.
- After you have created the RAD, you should have a good overall idea of what you are building.
- You still may not have decided on the design details (see the SDD for this!).

Project deliverable – the RAD, content

- Sketches or prototypes of the GUI interface (all windows) along with a description of how the user will navigate through the interface.
- A list of **the other roles assigned to each team member**, on the assumption that everyone is a developer first.

Scrum requirements

- Requirements are collected from project stakeholders (*"you" in your case!*).
- Spend some time to make sure you have considered all the features you need to have in your system.
- Requirements are split into:
  - Functional Requirements
  - Non-Functional Requirements.

User stories

- The SCRUM methodology likes to list requirements in the format:

As a [stakeholder], I want to [goal] so that [motivation].

- Stakeholder – for whom are we are building the application?
- Goal – what are we doing?
- Motivation – why are we doing it?

User stories, Cont.

- User stories need to be clear to everyone and verifiable. The latter means that you need to be able to confirm that you have met the requirements of the story.

- Focus on user needs, rather than on the solution or technology domain.

User stories, Cont.

- Examples of Functional Requirements:

User stories, Cont.

- Examples of Functional Requirements:

As a user, I want to be able to save my game progress so that I can re-start the game from where I left off.

As a level designer, I want to be able to use a text editor to design and edit levels for simplicity.

User stories, Cont.

- Examples of non-functional requirements:

As a user, I want to be able to save my game in less than 2 seconds, so I only have to wait a short time.

As a user, I want to be able to play this game on an Android device or on my Mac, so I can play it anywhere.

User stories, Cont.

- Examples of non-functional requirements:

User stories, Cont.

- Keep collecting user stories until you think you have covered everything.
- Separate them into functional and non-functional.
- Try to keep them from overlapping.
- Break overly large requirements up, if you can. Something overly large will be characterized by a **long time-to-completion that is difficult to estimate.**

User stories, Cont.

- The functional requirements will form the basis of your product backlog, expressed as issues on GitLab. Take advantage of GitLab milestones Feature.
- You will prioritize these and estimate time requirements. Take advantage of the labeling feature.
- Select some items from the backlog to form a sprint.

Non-Functional Requirement

- It is harder to come up with Non-Functional Requirements without having something already built.

Non-Functional Requirements

- More like a system specification. How to express these things?
  - One possibility: Create a task issue that is a test that specifies the requirement.
  - The task will likely be linked to a story that must be complete in order for the test to take place.
- A “failed” test will result in normal tasks or, if it is a higher priority, might result in a story of its own.

Non-Functional Requirements

- It might help to consider several “real-world” categories:

Non-Functional Requirements

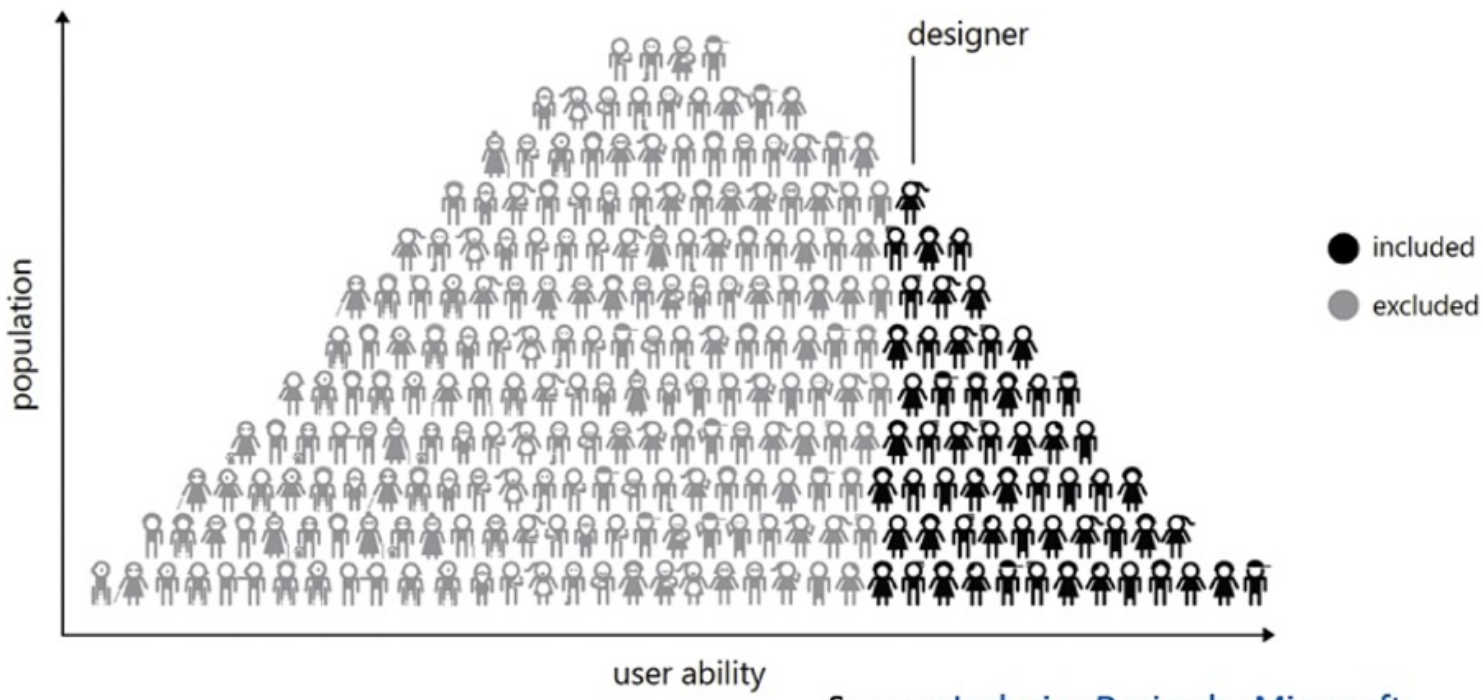
- It might help to consider several “real-world” categories (they might not all apply to you):
  - Usability
  - Reliability
  - Performance
  - Supportability
  - Implementation
  - Interface
  - Operation
  - Packaging
  - Legal

Non-Functional Requirements

Cognitive Walkthrough Methods
+
Personas

- <u>Usability</u>
  - What is the ability level of user?
  - What interface standards are already familiar to the user?
  - Affordances?
  - What documentation do they need? Paper, pdf, web, CD?
  - Will they use a help system or documentation *or are they an engineer*…

Usability

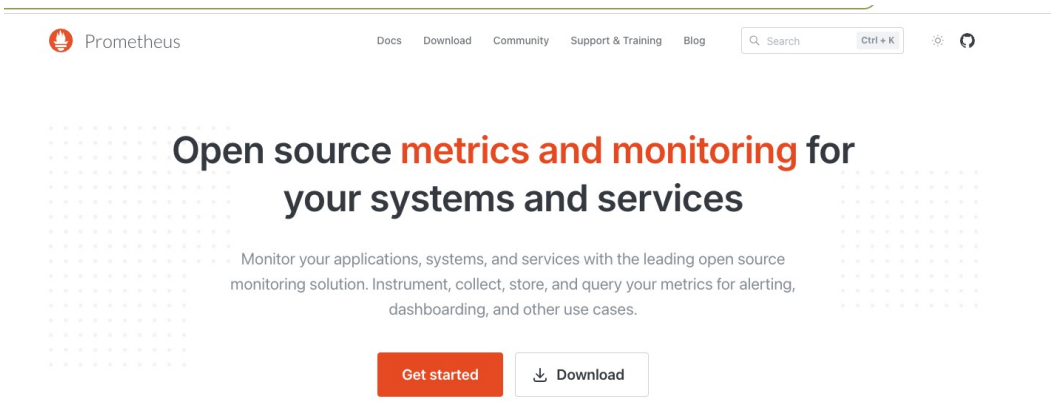


Source: Inclusive Design by Microsoft

Non-Functional Requirements

- <u>Reliability</u>
  - How reliable, available and robust?
  - Can it be restarted after a failure?
  - How much data can be lost?
  - How are exceptions handled?
  - Any safety requirements?
  - Any security requirements?

Evaluation window
Monitoring tools (e.g., Prometheus)



Non-Functional Requirements

- <u>Performance</u>
  - How responsive?
  - What user tasks are time critical?
  - How many concurrent users (now and in the future)?
  - How much data?
  - What is acceptable latency?

Benchmarking
Load testing

Non-Functional Requirements

- Supportability
  - What extensions will be needed?
  - Who does the maintenance?
  - Ported to different environments?

Non-Functional Requirements

- Implementation
  - Constraints on the hardware platform?
  - Constraints imposed by maintenance requirements?
  - Constraints imposed by testing team?

Non-Functional Requirements

- Interface
  - Required interface with existing system?
  - How is data imported/exported?
  - Existing standards already in use by client?

Non-Functional Requirements

- <u>Operation</u>
  - Who will manage the system when it is operating?
    - Third-party provider
      - The service-level agreement (SLA)
    - Internal Team
    - Collaborative Approach

Non-Functional Requirements

- Packaging
  - Who does the installation?
  - How many installations are anticipated?
  - How long can the installation take?

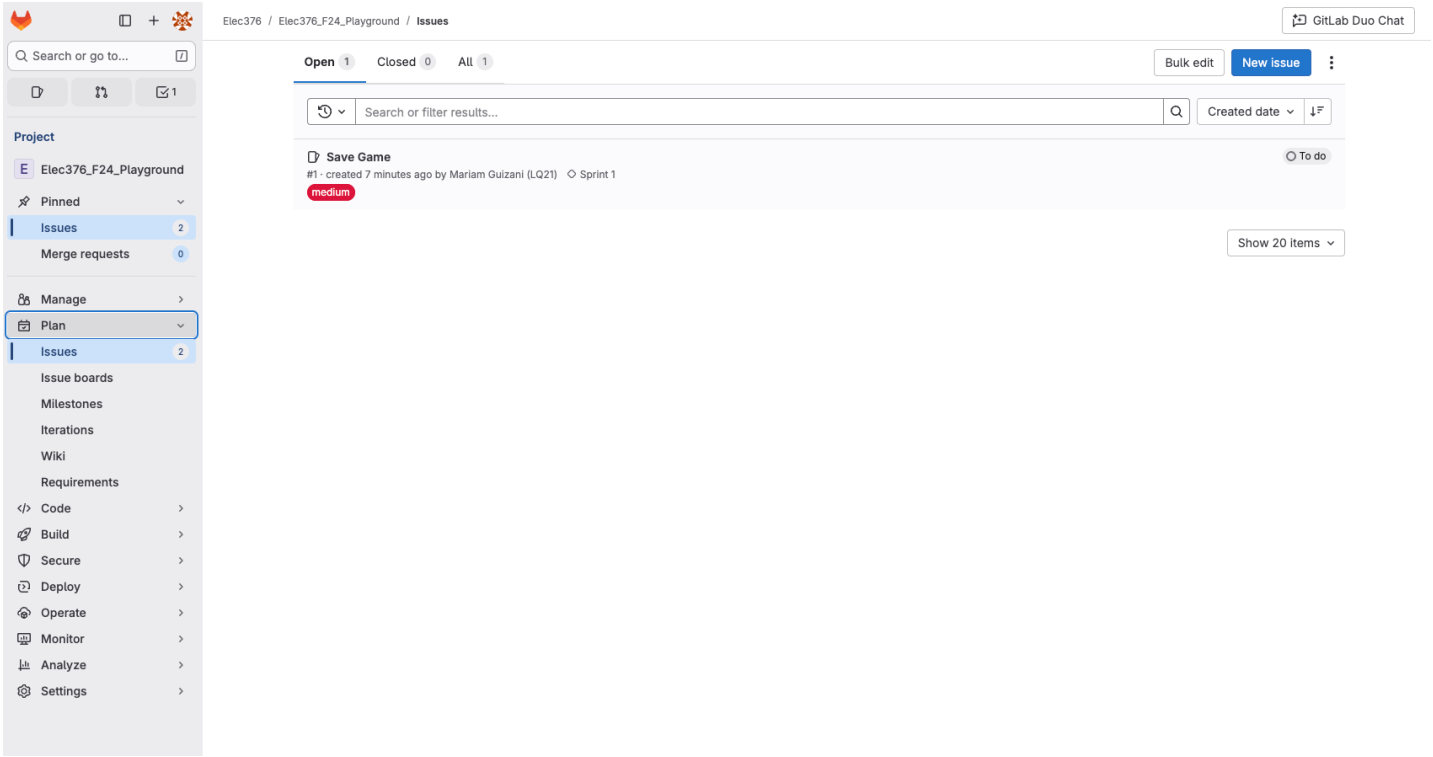
Non-Functional Requirements

- Legal
  - How is it licensed?
  - What liability results from system failures?
  - Does the use of any components incur royalties or licensing fees?

Non-Functional Requirements

- Only a few of these may apply to your project, but:
- Non-functional requirements will likely determine the **bounds of your system in terms of what can be done and what cannot**. They help define the **quality attributes and constraints** under which the system must operate.
- They will form design goals that everyone needs to keep in mind when they are implementing functional requirements.

>>


Elec376 / Elec376_F24_Playground / Issues
GitLab Duo Chat
Search or go to...
Project
Elec376_F24_Playground
Pinned
Issues 2
Merge requests 0
Manage
Plan
Issues 2
Issue boards
Milestones
Iterations
Wiki
Requirements
Code
Build
Secure
Deploy
Operate
Monitor
Analyze
Settings
Open 1
Closed 0
All 1
Bulk edit
New issue
Search or filter results...
Created date
Save Game
#1 · created 7 minutes ago by Mariam Guizani (LQ21) Sprint 1
medium
To do
Show 20 items

>>

RAD_Example1.pdf
Page 7 of 12

used devices

As a user, I want to interact with the program through the keyboard and mouse/on screen clicks for ease of use.

As a user, I want to receive output from the program within 3 seconds of creating an input through a click or a search

As an editor, I want to be able to update upcoming events, FAQ and student services information by editing a file that the program reads upon Startup, for ease and simplicity.

As an editor, I want to be able to update the program by re installing a new version of the program on the phone or PC to allow for revisions and improvements.

Sketches of the GUI interface

The campus map view

The user can filter by building destination type using a check box system to indicate what they would like to see on the map. Below are two sample sketches of what the map may look like: on the left is an unfiltered map view showing all building types, and on the right is a map view filtered per the user's request, to show, in this example, the dining halls, snack/ drink locations, and library buildings.