**GONZAGA UNIVERSITY**

**School of Engineering and Applied Science**

**Center for Engineering Design and Entrepreneurship**

**Medcurity Network Inventory**

**Project Risks, Release, Management, and Maintenance Plan**

**Sections 04-07**

**Release:**

**Draft v0.1**

**PROJECT PLAN DRAFT STAGE DOCUMENT**

**October 8, 2023**

**Medcurity Network Inventory Team**



**Brandon Huyck**

**Colleen Lemak**

**Artis Nateephaisan**

**Jack Nealon**

**4 Project Risks**

Provide a list of the major risks that are associated with your project. For each risk, you must clearly and concisely describe: (1) why it is a risk to the project (e.g., what will the potential impact be to the project); (2) what actions you will take to prevent the risk from happening; (3) how you will monitor the risk; (4) what events/situations will trigger the need to mitigate the risk (i.e., when will you know to switch to “plan B”); and (5) what you will do if the risk does becomes a reality (i.e., what is your “plan B”, “plan C”, etc.). It is not enough to just list your risks; you must also have a plan to prevent, monitor, and mitigate each risk.

1. Data corruption
   1. Impact: System data may become corrupted, resulting in inaccurate or unreliable records; functionality and performance may also be negatively affected.
   2. Prevention: Ensure backup procedures are in place, perform regular database maintenance, and implement secure authentication upon tool execution.
   3. Monitoring: Frequently check the database and its entries for potential inaccuracies, data loss, and unexpected program crashes.
   4. Event Trigger: Any type of suspicious activity or database error messages must be mitigated.
   5. Mitigation: Recover data lost from backups and analyze the source of corruption. Next, document proactive steps to prevent future loss.
2. Declining database performance
   1. Impact: When filtering and storing large sets of inventory data, database performance may decline over time.
   2. Prevention: Update selected database to the most recent version of its releases, and install the latest version of the host operating system to avoid preventable technical issues.
   3. Monitoring: Evaluate performance statistics, record query response time, and monitor for potential overuse of capacity.
   4. Event Trigger: Observable slow performance when retrieving information or lengthy freezing during requests would require mitigation.
   5. Mitigation: Ensure queries are optimized when sent to the database and remove any entries that are not needed.
3. Data security breach
   1. Impact: Malicious actors may gain unauthorized access to sensitive data, causing data loss and extensive privacy breaches.
   2. Prevention: Verify authentication is secure and usable, with a focus on what the visible scope is for each user.
   3. Monitoring: Communicate any irregularities, suspicious login attempts, unexpected situations, or security alerts.
   4. Event Trigger: Be aware of any security alerts or unusual activity on local host machines or git repository files.
   5. Mitigation: Notify stakeholders of issues and contain systems that interact with the tool inventory system.
4. Software compatibility
   1. Impact: Performing scans of networks may not return all software devices due to compatibility issues, leading to inaccurate data or uncapturable data.
   2. Prevention: Install the most recent software for the database and ensure software between user interface, database, and authentication is compatible with various tests to verify proper connectivity.
   3. Monitoring: Especially in developmental stages, monitor inventory data for inconsistent entries or inaccurate records.
   4. Event Trigger: If inaccurate .csv reports are commonly generated, issues likely involve software compatibility.
   5. Mitigation: Test libraries and any potential crawler agent 3rd party tools early and often to verify accuracy and usability within project scope.

**5 Initial Product Release Plan**

**5.1 Major Milestones**

**Table 3: Major Milestones**

| *Milestone* | *Description* | *Target Completion Date* |
| --- | --- | --- |
| *DatabaseDesign* | The database design should be complete, so we can start coding it for itself and the connecting parts. This needs to be available for review before the end of the first semester to provide time for revisions. | First week of December |
| *CrawlerAgentOutline* | Specifics of the crawler agent’s implementation (e.g. build from scratch or wrap open-source, how it will interact with the user and network, what information it will collect, how it will connect with the database, etc.) should be mapped out to facilitate success in the second semester | First week of December |
| *TestingPlan* | Decisions on how the various pieces of the software will be tested should be decided. This should include what parts need unit testing, how to go about user-acceptance testing, etc. | First week of December |
| *UserInterfaceLayout* | The user interface should be fully sketched out, allowing for approval before starting to implement it in the second semester. | First week of December |
| *DatabaseVersionZero* | Database will be built siloed. It will be hosted on a test AWS server and filled with test data emulating real-world conditions | First week of March |
| *CrawlerAgentVersionZero* | Crawler agent will be built siloed. It will be functional on a test network, traversing it, seeing all targets, and collecting appropriate data | First week of March |
| *TestingUnderway* | Testing will be complete to various extents, with multiple unit tests written for each component, and user-acceptance testing and integration tests planned out with extensive details | First week of March |
| *UserInterfaceVersionZero* | Each component of the user interface will be built siloed, without necessarily being connected together. It will not be connected to the database or other components of the project. Feedback from stakeholders will be requested and inform the final stage of its development | First week of March |
| *TestingComplete* | All testing will be in place, completed, and passing | Second week of April |
| *ComponentIntegration* | The database, crawler agent, and user interface will be connected, communicating with each other, and are functional in a development environment emulating production. | Second week of April |
| *DocumentationComplete* | Documentation on each individual component, testing results, future directions, the software as a package, and end-user instruction will be written and saved in the repository | Third week of April |

**5.2 Initial Sprint Releases**

Provide an initial plan for the work you will complete and demo at the end of each sprint for the entire year. Your initial sprint plan should connect the major milestones with your project backlog and major features. Thus, e.g., your release plan should coincide with your project milestone schedule above.

For each sprint, state what requirements will be worked on and concretely what you will demo for your target users and sponsor at the end of the sprint. Note that you will need time in your release plan for system and usability testing as well as system deployment and documentation. Be sure to clearly state when usability testing, deployment, and documentation will be performed.

Summarize your sprint plan in a table as follows. Note that it is expected that the sprints for the rest of this semester are more precise than those towards the end of the year and that your plan may change. However, you must demonstrate that you have spent time thinking about the high-level release plan for your project.

**Table 4: Sprint Release Plan**

| ***No*** | ***Sprint Date*** | ***Sprint Length*** | ***Sprint Goal*** | ***Backlog*** | ***What we will demo*** |
| --- | --- | --- | --- | --- | --- |
| ***1*** | ***10/24-11/6*** | **14** | **Evaluate how we handle the first sprint and to get initial designs done.**  **Complete ~1/3 total development on Database design, Crawler Agent Outline, Testing Plan, User Interface layout** |  |  |
| ***2*** | ***11/7-11/20*** | **14** | **Complete ~3/4 total development on Database design, Crawler Agent Outline, Testing Plan, User Interface layout** |  |  |
| ***3*** | ***11/27-12/4*** | **8** | **Database design, Crawler Agent Outline, Testing Plan, User Interface layout Due, Create second presentation** |  |  |
| ***4*** | ***12/5-12/18*** | **14** | **Dead week + finals, presentation on 12/6, final project plan due 10/19** |  |  |
| ***5*** | ***12/19-1/1*** | **14** | **Winter Break sprint 1** | **Depending on schedules, work will vary** | **Depending on schedules, work will vary** |
| ***6*** | ***1/2-1/15*** | **14** | **Winter Break sprint 2** | **Depending on schedules, work will vary** | **Depending on schedules, work will vary** |
| ***7*** | ***1/16-1/29*** | **14** | **Complete ~1/4 total development on Database Version Zero,**  **Crawler Agent Version Zero,**  **Testing Underway,**  **User Interface Version Zero** |  | **Current Software Versions** |
| ***8*** | ***1/30-2/12*** | **14** | **Complete ~2/4 total development on Database Version Zero,**  **Crawler Agent Version Zero,**  **Testing Underway,**  **User Interface Version Zero** |  | **Current Software Versions** |
| ***9*** | ***2/13-2/26*** | **14** | **Complete ~3/4 total development on Database Version Zero,**  **Crawler Agent Version Zero,**  **Testing Underway,**  **User Interface Version Zero** |  | **Current Software Versions** |
| ***10*** | ***2/27-3/10*** | **13** | **Database Version Zero,**  **Crawler Agent Version Zero,**  **Testing Underway,**  **User Interface Version Zero Due** |  | **Current Software Versions** |
| ***11*** | ***3/18-3/25*** | **8** | **Sprint after spring break to get back into the project. Start documenting version 0 bugs and where the software is lacking.** | **Will vary on what isn’t working.** | **Documentation on what needs to be fixed.** |
| ***12*** | ***3/26-4/8*** | **14** | **Refine and optimize version zero software** | **Will vary on what isn’t working.** | **Optimized software components.** |
| ***13*** | ***4/9-4/22*** | **14** | **Final sprint before delivery week. Finalize database, crawler agent, testing, and UI** | **Will vary on what isn’t working.** | **Refined and optimized versions of project component** |
| ***14*** | ***4/23-5/3*** | **11** | **Design Expo, Presentation #4 and Senior Celebration Events all on 5/1, final report due 5/3** | **Presentation and FINAL tweaks to software** | **Deliver the project** |

**6 Maintenance Considerations**

Provide a brief description of maintenance issues regarding the system you develop. In particular, discuss whether there is an identified group that will provide maintenance of the system (and what their level of expertise is), what parts of the system may require maintenance, how you plan to develop the system to support future maintenance, what level of expertise will be required to perform maintenance on the system, and any other issues concerning maintenance of the system you develop.

Some aspects of our project that should be monitored throughout the product’s lifespan are the database and its performance, security issues, and compatibility with future software. We expect that the IT department of Medcurity, the most likely to be experienced with this line of work, will be able to, with the use of our thorough documentation, be able to understand and maintain all aspects of the product in the future.

While the database should be fairly simple and straightforward, there is a chance that issues may arise if invalid entries for the database appear or improperly formatted data. Tests will be developed to account for these issues. Should Medcurity seek to change fundamental aspects of the database, such as what users are allowed to enter or how the data should be formatted, the IT department should be able to clearly find where they should be able to do that. If the performance of the database dips, there should be systems in place for the IT department to evaluate performance statistics and pinpoint where the issue may be.

Security issues should be accounted for with a verification system for the product and should remain the same throughout the product’s lifespan. If Medcurity wants to create additional fields in authentication to ensure user-sensitive data is secure, the documentation should allow the IT department to seemingly add more forms of security for the system.

Software updates could potentially render our system incompatible or outdated as time goes on. In order to prevent this, all sections of the code should be clearly and thoroughly documented. The IT department should be able to understand every section’s logic and code through documentation, allowing them to modify it should the time come.

**7 Project Management Considerations**

Provide a brief description of how you plan to organize yourself as a team to complete the

project deliverables. Minimally, this should include where and when your weekly team meetings will be, when/where you will meet with your project sponsor (for standing meetings), and how you will update your sponsor, primary DAB member, faculty advisor, and other stakeholders of your progress. Also include a description of how you are planning on breaking up the work for your project among team members. Each team member must be responsible for some concrete aspect of the system. Finally, list any additional tools you plan to use to communicate your progress and/or elicit feedback from your sponsor, liaison, and end users.

Our team plans to meet every Tuesday at 4:30 p.m. at PACCAR 203. Our faculty advisor will accompany the team in most meetings, and our primary DAB member and sponsors will attend whenever they must be present for discussion or updates. We have been using email as our primary form of communication, but everyone has joined a Discord server where brief communication can be made. We will plan to give our sponsor, DAB member, faculty advisor, and relevant stakeholders frequent updates of our progress, potentially weekly or bi-weekly, allowing them to be informed about details of the project. This is how we plan to break up the project, but it is not set in stone yet:

Brandon Huyck: Network Crawler Agents

Coleen Lemak: Database

Jack Nealon: User Interface

Artis Nateephaisan: Testing

We plan to use email to give progress reports, schedule meetings, ask questions, and receive feedback from liaisons and end users. If email is unnecessary for things such as quick questions, people may also send messages through Discord as an easy and informal way of communication.