**….Risk Management Technique…..**

**1) Risk table**

| Risk | Category | Probability | Impact |
| --- | --- | --- | --- |
| Server Crash | Project Risk | 15% | 4 |
| Data security threats | Technical Risk | 30% | 4 |
| Requirements change | Project Risk | 25% | 1 |
| Mailing failure | Technical Risk | 10% | 2 |
| API unreachability | Technical Risk | 20% | 2 |

Impact Values

1 - Catastrophic

2 - Critical

3 - Marginal

4 – Negligible

**2) Risk management methods**

**RISK 1: SERVER CRASH**

MITIGATION: The cost associated with server crash is about the dissuading of returning customers from regularly using the web app. A server crash leads to the disruption in the notification system of the web app leading to disruption in the user’s daily usage and mass user disappointment. To overcome this, the notification system could be dispatched separately on cloud schedulers.

MONITORING: The developers should be focused on regularly pinging the notification systems to check the server status and verify whether the system has crashed.

MANAGEMENT: The developers should try to maintain notification APIs separate from the web server to ensure smooth,continuous running of the services.

**RISK 2: Data Security Threats**

MITIGATION: Databases are always under threat of various cyber attacks from malicious users. As such, data can be breached leading to theft of sensitive user data or the notes and tasks being maintained by the user. This is a major violation of user privacy and could lead to wanton or even jeopardous consequences for users. To overcome this, multi-level encryption algorithms could be used.

MONITORING: The developers should be focused on enforcing security parameters in the user authentication modules to ensure that any unmediated access is logged and reported.

MANAGEMENT: The developers should try to maintain multi-level encryption and latest cybersecurity technologies.

**RISK 3: REQUIREMENTS CHANGE**

MITIGATION: In order to prevent this, regular meetings should be held with the stakeholders (preferably using agile framework) to discuss and validate the requirements

MONITORING: The developers should be focused on repeatedly performing validation tests with the stakeholders using forms,questionnaires and meetings.

MANAGEMENT: The developers should try to repeatedly use the new feedback received from validation tests in the next iteration of the product development process.

**RISK 4: Mailing failure**

MITIGATION: Mailing failures are pretty common whenever different sorts of mailing proxy servers are used by the product.

This can lead to a disruption in the notification system and users missing their deadlines. This will become a huge source of disappointment. This can be avoided by using multiple backup proxies.

MONITORING: Pinging mailing servers regularly can help to check the status of mailing servers used.

MANAGEMENT: A chain of backup servers to be used in case of primary mailing server failures should be prepared to prevent this risk from becoming reality.

**RISK 5: API failure**

MITIGATION: API failures are pretty common whenever different sorts of APIs are used by a product for functionalities.

In Noteffy’s case,it can be the chart and external mailing functionality. This can disrupt the user’s use flow. This can be avoided by using error handling mechanisms and alternative user screens

MONITORING: Pinging APIs at the start of each user session can help to smoothen the user experience.

MANAGEMENT: A horde of alternative error pathways and user screens can be used to handle inaccessible or non-servicing APIs. Maintaining and utilising a list of similar APIs can also be productive.