CampusTalk

Risk Mitigation, Monitoring and Management Plan

Introduction

The goal of the risk mitigation, monitoring and management plan is to identify as many potential risks as possible. For identification of risks involved in developing CampusTalk, a checklist is prepared containing risks, impact level and priority of impact.

Once all risks are identified, each risk will be provided with mitigation, monitoring and management plan. The quicker the risks can be identified and avoided, the smaller the chances of having to face that particular risks's consequences. The fewer consequences suffered as a result of good RMMM plan, the better the product, the smoother the development process.

- **Technical Issues**
  Issue that mainly pertains with the tools or technologies used for the project. It may be an issue that affects the system used, or may a lack of knowledge by team member regarding technology.
- **Business Impact Risk**
  This type of risk mainly affects the business/monetary aspect of the project development, and thus primarily affects the stakeholders of the project.
- **Consumer Related Risk**
  These risks affect the user-base of the system, the direct consumer of the product, thus indirectly affects the business aspect of the project as well.
- **Process Risk**
  This risk a hinderance in the development process of the project, but affected one or many of the processes involved in the project development.
- **Development Risk**
  These risks affect the development of the project itself, they may be technical and more closer to development process itself.

Risk Table

Risks	Category	Probability	Impact
Computer Crash	TI	60%	2
GitHub Inaccessible	DE	10%	1
Lack of Java EE Knowledge	DE	80%	2
Changes in Requirements	PR	10%	2
Lack of Development Experience	CU	60%	2
Lack of Database Storage/Availability	TI	40%	2
Poor Documentation	DE	30%	3
Deviation from SEN Standards	DE	50%	2
Late Delivery	BU	30%	2
Matching Product with other Team	BU	80%	3

Category Legends:

- TI – Technical Issues
- BU – Business Impact Risk
- CU – Consumer Related Risk
- PR – Process Risk
- DE – Development Environment Risk

Impact Legends:

- 1 – Catastrophic
- 2 – Critical
- 3 – Marginal
- 4 – Negligible

Risk : Computer Crash

- **Mitigation**
  The cost associated with crashing a computer has more to do with the data lost rather than computer itself. While entire code and critical documents are synced on the internet and thus data lost here is neglible, but what makes this risk *critical* is the fact that development environment configured in the computer is lost until it is fixed or recovered.

- **Monitoring**
  While working on the project, members should make sure that project source code synced and commits are up-to-date with GitHub.com (our primary medium of sharing code). Also, all the documents created are regularly sycned with Dropbox and all team members have latest copies of documents.

- **Management**
  Lack of stable computing environment is critical issue for developers, and in the era of cloud computing, it is a good idea to keep every important document and file on the *cloud* such that it is always available and safe from data lost.

Risk : GitHub Inaccessible

- **Mitigation**
  To share source code accross team, and keeping track of every change, third-party service **GitHub** is used, hence inaccessiblity to the service due to some reason may lead to zero communication with team members regarding source code if team members are at geographically distant locations, and hence this may lead of slower development.

- **Monitoring**
  Along with copy of source code on GitHub, team members are encouraged to keep up-to-date copy of code repository in local machine, and use different mediums of sharing the code such that when GitHub is back up and running, nothing needs to be started all over again.

- **Management**
  Sharing and tracking changes code in a larger team is crucial and GitHub provides best way to do that with using Git as an underlying technology of version control. Hence, even if the chances for the service to be inaccessible are very low, it may still occur and members must be prepared with a string backup plan.

Risk : Lack of Java EE Knowledge

- **Mitigation**
  CampusTalk is complete Java web application and thus uses Java EE (Servlet/JSP) technology on the server-side of the application, thus it is critical for developers to have fairly good knowledge of the platform. Lack of such may lead to slower developement and lack of contribution in development from the members who might be good programmers but are just unaware of the platform used in the development of the product, and thus overall product gets affected due to its single-handed development.

- **Monitoring**
  During the research, feasibility and requirements collection phase of the project, team members are encouraged to collect as much information as possible regarding the platform to be used for development, and learn it before the actual development starts such that they don't spend much time in learning during development.

- **Management**
  Choice of development environment large depends on the requirement of the system, and it is not always possible that all the team members are aware of the technologies used in the entire project and hence there's a learning curve always involved during project development. Thus, its responsibility of the team member to learn the required tools and technologies in prior before development starts.

Risk : Changes in Requirements

- **Mitigation**
  In order to prevent this, we should analyze requirement properly, and conduct meeting with team member on regular basis. In our project , chances of change in requirement will be less as we don't have specific client to deal with. And we have decided what we will going to deliver at end.

- **Monitoring**
  The meetings with the Team should ensure that our team members understand each other and the requirements for the project.

- **Management**
  We are using the Github (service based on Git Version Control system) for managing any changes.

Risk : Lack of Development Experience

- **Mitigation**
  In order to prevent this from happening, the development team will be required to learn the languages and techniques necessary to develop this software. The member of the team that is the most experienced in a particular facet of the development tools will need to instruct those who are not as well versed.

- **Monitoring**
  Each member of the team should watch and see areas where another team member may be weak. Also if one of the members is weak in a particular area it should be brought to the attention by that member, to the other members.

- **Management**
  The members who have the most experience in a particular area will be required to help those who don't out should it come to the attention of the team that a particular member needs help.

Risk : Lack of Database Storage/Availability

- **Mitigation**
  In order to prevent this from happening, developers who are in contact with the database, and/or use functions that interact with the database, should keep in mind the possible errors that could be caused due to poor programming/error checking. These issues should be brought to the attention of each of the other members that are also in contact with the database.

- **Monitoring**
  Each user should be sure that the database is left in the condition it was before it was touched, to identify possible problems. The first notice of database errors should be brought to the attention of the other team members.

- **Management**
  Should this occur, the team would call a meeting and discuss the causes of the database instability, along with possible solutions.

Risk : Poor Quality Documentation

- **Mitigation**
  In order to prevent this from happening, members who are in charge of developing the documentation will keep in contact with each developer on the team. Meetings will be held routinely to offer documentation suggestions and topics. Any topic deemed missing by a particular developer will be discussed and it will be decided whether or not to add that particular topic to the documentation. In addition, beta testers will be questioned about their opinion of the documentation.

- **Monitoring**
  Throughout development or normal in and out of house testing, the development team and or beta testers will need to keep their eyes open for any possible documentation topics that have not been included.

- **Management**
  Should this occur, the team would call a meeting and discuss the addition of new topics, or removal of unnecessary topics into the documentation.

Risk : Deviation from SEN Standards

- **Mitigation**
  While it is possible to deviate from software engineering standards, it is unlikely to occur. All team members have a full understanding of the software process, and how we plan to implement them in the process.

- **Monitoring**
  Technical reviews involving comparison between documentation and the actual project will help to determine if deviation will occur. All relevant documents must be as complete and accurate as possible to ensure that work will conform to expressed software engineering standards.

- **Management**
  Should deviation occur, steps must be taken to guide the project back within the standards expressed in accompanying documents. Technical reviews help to determine what must be done to keep the project in line with established software engineering standards.

Risk : Late Delivery

- **Mitigation**
  The cost associated with a late delivery is critical. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

- **Monitoring**
  A schedule has been established to monitor project status. Falling behind schedule would indicate a potential for late delivery. The schedule will be followed closely during all development stages.

- **Management**
  Late delivery would be a catastrophic failure in the project development. If it becomes apparent that the project will not be completed on time, the only course of action available would be to request an extension to the deadline form the Customer.

Risk : Matching Product with Other Team

- **Mitigation**
  If Two Team’s are working on the Same product then there’a competition risk. We have to compare ourself with theirs. Risk of designing our documents and project more good comparision to them. So, Providing less facility and lack of user interface is a risk.

- **Monitoring**
  If possible, try to make comparision beween team developing same product. E.g, Which new features are they’re providing, can we provide better then them? Monitor their Team.

- **Management**
  We can provide better interface, faster performance than other team’s by using latest technologies and tools. Providing more features compared to them.

***