Hacettepe University
Department of Computer Engineering
BBM478 - Software Engineering Laboratory

Risk Management Report

Group One

Ahmed Şamil BÜLBÜL (21426749, Project Manager)
Halil İbrahim ŞENER (21328447, Software Developer)
Naciye GÜZEL (21580841, Tester)

1. Risk Management Specifications In Deliverable 3

Risks forecasted in planning	How it was handled/mitigated
1. Integration problems	By using Travis CI.
2. Availability of software / Operating system dependency problems	By using Java, we eliminate most of the problems.
3. Availability of database in different systems problem	By using MySQL, the database will be available in most of the systems.
4. Communication channel problems	By using Slack, we created a distraction-free environment.
5. Build dependencies and build portability problems	By using Maven.
6. Development complexity increases due to increase in problem and feature sizes.	By using specific branches for development and Travis CI.
7. Time to learn programming language	By using Java we have prevented this conflict.

1. Integration problems

Integration is a big problem when we are working on different features. We create branches to create features then merge them. The problem here is developing multiple features. When we create multiple branches and done working with them, we try to merge them to master branch. This will create problems on merging. Also after long times in development process, it will be harder to integrate. To prevent this, we need to check that whole system is working and it is not causing errors. By using Travis CI and periodic integration we prevent this problems.

2. Availability of software / Operating system dependency problems

We are providing this system to all operating systems. It is hard to develop product for every system natively. To solve this problem, we decided to use Java which is available in all systems.

3. Availability of database in different systems problem

We are providing this system to all systems. In development process we are trying to use a database that is available in all systems. To solve this problem, we decided to use MySQL which is available in all operating systems.

4. Communication channel problems

We need a communication channel to discuss, overview project and create a good environment for team members. It should be disturb-free environment, specific for development process. To solve this problem, we decided to use Slack which is available on both desktops and mobiles.

5. Build dependencies and build portability problems

Throughout development process, we are using different environments such as different operating systems to satisfy requirements. Different APIs, libraries might be requires. This might create dependency and build issues. To solve this problem, we decided to use Maven.

6. Development complexity increases due to increase in problem and feature sizes.

We are developing a system that gets more complex overtime. Features should be clearly specified and a feature that is not completed or wrong should be checked. To solve this problem, we decided to use feature branches and Travis CI to integrate them.

7. Time to learn programming language

Programming language time might slow down development process and inexperienced team members can create bad code. To solve this problem, we decided to use Java that every member knows well.

Risks NOT forecasted in planning, but observed	**How it was handled/mitigated**
Direct commits to master branch which leads to disorder and codes with unpredicted bugs.	Branch protection is used.

1. Direct commits to master branch which leads to disorder and codes with unpredicted bugs.
While developing the project, we saw that we work collaboratively while developing features and creating documents. However after developing some features, we encountered problem of direct commits to master branch which might create conflicts. To solve this problem, we started to use branch protection.

2. Risk Management Specifications In Deliverable 5

Risks forecasted in planning	**How it was handled/mitigated**
1. Merge conflicts	By using GitHub conflict resolver.
2. Using different versions of tables in MySQL database	By creating a dump of database
3. Learning MySQL	By using a common approach for database statements and studying tutorials.
4. Database access	By using Database Access Objects.

1. Merge conflicts are sometimes inevitable. We were working on different branches to create different features. This was helping us to stop editing files in workspace of other features. Still we couldn't stop some conflicts. Throughout project, every conflict we have encountered was small conflicts, so we used GitHub's website to resolve them.

2. Even though we were using same database, one change in database of one of the team members should affect others. This is done by using MySQL dump. Dump helps us to share changes in database and create a common basis for database.

3. We had to learn SQL in order to create a database and provide functionality to application. But everybody didn't start learning from scratch. Everybody started to do what they know best and shared information then we created a common knowledge about database. Since database connections are done using JDBC it also made things simpler to learn.

4. Sometimes, for specific features that is completely independent from other features, database accesses becomes complex. We cannot put everything in one class. So we created database access objects in order to make things less complicated. Still not every database access is in a DAO.

Risks NOT forecasted in planning, but observed	**How it was handled/mitigated**
1. Time management	By re-arranging schedule

1. It is not always possible to spare time to project. We tried divide features according to team member's schedule but it was not always possible. So we gave everybody the amount of work they can handle.