|  |  |
| --- | --- |
| Choose the wrong technology at this stage | likelihood \* Impact = Exposure  0.2 \* 5 = 1.0 |
| Miscommunication may happen between manager and developers | 0.4 \* 5 = 2.0 |
| Miscommunication may happen between customers and developers | 0.4 \* 8 = 3.2 |
| Unclear requirement and omit requirement may happen | 0.4 \* 5 = 2.0 |
| Other risks may lead the underestimate of developing period | 0.4 \* 2.5 = 1.0 |

Risks in requirement analysis:

Risks in Design:

|  |  |
| --- | --- |
| Wrong data structure / design pattern chosen | likelihood \* Impact = Exposure  0.2 \* 8 = 1.6 |
| Poor organizational structure and definition of roles and responsibilities | 0.2 \* 5 = 1.0 |
| Design errors and changes | 0.2 \* 5 = 1.0 |
| Other risks in this phase may lead to delay of design and permits | 0.7 \* 2.5 = 1.75 |
| Improper choice of the programming language or frame (technology part) | 0.1 \* 5 = 0.5 |

Risks in Coding:

|  |  |
| --- | --- |
| Very intense schedule | likelihood \* Impact = Exposure  0.7 \* 2.5 = 1.75 |
| Insufficient resources | 0.3 \* 5 = 1.5 |
| Continuously changing requirements | 0.7 \* 5 = 3.5 |
| Employee turnover | 0.5 \* 5 = 2.5 |
| Numerous bugs during the coding | 0.3 \* 8 = 2.4 |

Risks in Deployment

|  |  |
| --- | --- |
| Bad testing may lead to potential bugs | likelihood \* Impact = Exposure  0.3 \* 8 = 2.4 |
| Product may have issue when deploy to target platform | 0.2 \* 8 = 1.6 |
| Conflicting user requirements | 0.4 \* 5 = 2.0 |
| Ignoring the nonfunctional requirements | 0.4 \* 5 = 2.0 |
| Missing capabilities | 0.4 \* 2.5 = 1.0 |

Risks in maintenance:

|  |  |
| --- | --- |
| Employee turnover | likelihood \* Impact = Exposure  0.4 \* 2.5 = 1.0 |
| Changing requirements | 0.6 \* 2.5 = 1.5 |
| Over budget | 0.4 \* 2.5 = 1.0 |
| Change in environment | 0.2 \* 2.5 = 0.5 |
| Problem with bug fix | 0.4 \* 5 = 2.0 |

Top five risks are:

Miscommunication between customers and developers in requirement analysis (Exposure 3.2)

* Mitigation plane – use frequent communication rather than one time and make sure several developers involve the discussion not one person.
* Monitoring strategy – deliver short term product to customers to make sure the function is what they ask

Continuously changing requirements in coding phase (Exposure 3.5)

* Mitigation plane – there is no way to avoid changing requirements but have to get used to it
* Monitoring strategy – discuss frequently to avoid misunderstanding by changing requirements

Numerous bugs during the coding in coding phase (Exposure 2.4)

* Mitigation plane – hire experienced developer and well test the code
* Monitoring strategy – spot the bug in time and fix it

Bad testing may lead to potential bugs in deployment phase (Exposure 2.4)

* Mitigation plane – hire experienced tester and use well designed test case
* Monitoring strategy – use well designed test case to guide the tester along the development

Employee turnover during coding phase (Exposure 2.5)

* Mitigation plane – use backup developers and communicate with employee in time
* Monitoring strategy – if the employee wants to leave it always a sign there to show he is going to leave, talk them and hire others in advance