সফটওয়্যার বাগ
কারণ প্রতিকার পরিণতি

অনিমেষ চন্দ্র বাইন

বাগ হলো কমপিউটার প্রোগ্রাম বা সিস্টেমের ত্রুটি বা খুঁত, যা সফটওয়্যারের প্রত্যাশিত ফল পেতে বাধাগ্রস্ত করে। আবার কখনও কখনও এমন ফল প্রদর্শন করে, যা প্রত্যাশিত ফলের সাথে কোনো মিল পাওয়া যায় না। আবার কখনও প্রত্যাশিত ফল এলেও প্রোগ্রাম সোর্সকোডে এমন কিছু সমস্যা থাকে, এর ফলে দীর্ঘ মেয়াদে বড় ধরনের ক্ষতির সম্ভাবনা থাকে। মূলত তিনটি বিশেষ কারণে এ ধরনের সফটওয়্যারের বাগ সৃষ্টি হয়। ০১. প্রোগ্রামারদের মাধ্যমে, ০২. আর্কিটেকচারাল ডিজাইনে ও ০৩. প্রোগ্রামের সোর্সকোড থেকে। তবে এর কম্পাইলারের মাধ্যমে খুব কম ভুলই সংঘটিত হয়।

অ্যাপ্লিকেশন বা সফটওয়্যারে বাগের জন্য অনেক প্রতিষ্ঠানকে প্রতিবছর বড় রকমের ক্ষতির মুখোমুখি হতে হয়, এর হাজারো উদাহরণ দেয়া সম্ভব। তবে এই লেখায় সফটওয়্যার বাগ, টুল এবং এর কৌশলগত দিক নিয়ে কিছু তথ্য উল্লেখ করা হলো। একই সাথে বিভিন্ন সময়ে সংঘটিত বড় ধরনের ক্ষতির দিকসমূহ তুলে ধরা হলো। এতে প্রোগ্রামাররা তাদের কাজের প্রতি আরও বেশি যত্নবান ও মনোযোগী থাকবেন।

১৯৬২ সাল। অনেক গবেষণার পর নাসার গবেষকেরা মেরিনার-১ নামের নভোযানটি মঙ্গলগ্রহের উদ্দেশে পাঠানোর সব ধরনের প্রস্তুতি শেষ করেন। কিন্তু উৎক্ষেপণের মাত্র কয়েক মিনিটের মধ্যে বহনকারী রকেটটি ধ্বংস হয়ে যায়। বিপর্যয়পরবর্তী গবেষণায় দেখা যায়, যে ত্রুটির জন্য এ বিপর্যয় সংঘটিত হয়, তা খুবই সামান্য। তবে ক্ষতির পরিমাণ অকল্পনীয়। প্রোগ্রামে এ ধরনের ভুল যে খুবই ধ্বংসাত্মক, তা বলার অপেক্ষা রাখে না। আর যা ঘটেছিল প্রোগ্রাম এরর তথা বাগের কারণে।

গবেষকেরা দেখতে পান প্রোগ্রামাররা যে প্রোগ্রাম লিখেছেন সেখানে বেগ পরিমাপক কোড হিসেবে 'R' ব্যবহার করা হয়। মূলত এটি হবে '$\bar{R}$' (বার)। উল্লেখ্য, 'R' হচ্ছে সাধারণ বেগ পরিমাপক কোড, কিন্তু '$\bar{R}$' হচ্ছে সুষম পরিমাপক কোড। প্রোগ্রামাররা ভুলবশত 'R'-এর মাথার ওপর একটি দাগ (হাইফেন) ভুল করে ব্যবহার করেন। আর এই দাগটি একটি মহামূল্যবান হাইফেন হিসেবে ইতিহাসে জায়গা করে নিয়েছে।

ইতিহাসের আরেকটি দুর্ঘটনা ঘটে ১৯৯৮ সালে খুব ছোট ভুলের জন্য। Mars Climate Orbiter সফলভাবে মঙ্গলগ্রহে অবতরণের অপেক্ষায়। কিন্তু মঙ্গলগ্রহের বায়ুমণ্ডলে প্রবেশ করার সাথে সাথে ধ্বংস হয়ে যায় নভোযানটি। অনুসন্ধানে বেরিয়ে আসে, এর অভ্যন্তরীণ বিভিন্ন কাজের সমন্বয়ের জন্য যে সফটওয়্যার ব্যবহার করা হয় সেখানে 'English units' (পাউন্ড সেকেন্ড)-এর পরিবর্তে 'Metric units' (নিউটন সেকেন্ড) ব্যবহার করা হয়।

একথা নিঃসন্দেহে বলা যায়, ওপরের দুটি ভুল খুব সামান্য হলেও এর জন্য ক্ষতির পরিমাণ অনেক বেশি। আর এর প্রতিটিই ঘটেছে সফটওয়্যার বাগের জন্য। মহাকাশ গবেষণায় এই সামান্য ভুলের জন্য প্রতিষ্ঠানের ৩২৭.৬ মিলিয়ন ডলারের প্রজেক্ট এক মুহূর্তে ধ্বংস হয়ে যায়।

কমপিউটার সফটওয়্যারে পাওয়া প্রথম বাগ

একথা সত্য, সময়ের পরিবর্তনের সাথে সাথে সফটওয়্যারের কোডিং ও পরীক্ষা-নিরীক্ষাপ্রযুক্তি উন্নত হয়েছে। ভুলও হচ্ছে। আর বড় ধরনের বিপর্যয় সৃষ্টি হচ্ছে যথারীতি।

এই উদাহরণটি মনোযোগ দিয়ে খেয়াল করলে একটি ছোট ভুল পাওয়া যাবে।

```
if ((scale !=0.0) || (scale !=1.0))
imageSize *= scale;
```

এর অর্থ হচ্ছে, যদি স্কেলের মান 0.0 অথবা 1.0-এর সমান না হয়, তবে স্কেল ফ্যাক্টর অনুসারে ইমেজ রিসাইজ করে। এই লজিকটিতে আসল ভুল হচ্ছে বুলিয়ান প্রকাশ পদ্ধতির। এখানে অথবা (OR)-এর পরিবর্তে এবং (AND) হওয়া উচিত ছিল। উল্লিখিত শর্তে ভুল থাকলেও প্রোগ্রামটি ঠিকই কাজ করছে, তবে দুটি শর্ত একই সাথে মানা হয়নি। শুধুই সাধারণ এ ভুল কোড অংশের জন্য ঘটেনি এ কথা সবাই বুঝতে পারবেন। এরকম হাজারো ভুল পাওয়া যাবে প্রোগ্রামারদের সোর্সকোডে। এখন প্রশ্ন হচ্ছে, এর জন্য কী করা যেতে পারে?

এনআইএসটি তথা ন্যাশনাল ইনস্টিটিউট অব স্ট্যান্ডার্ডস অ্যান্ড টেকনোলজির ইউএস ডিপার্টমেন্ট অব কমার্সের জন্য প্রস্তুত করা আরটিআইয়ের এক গবেষণা তথ্য থেকে জানা যায়, প্রতিবছর সফটওয়্যারে বাগ তথা প্রোগ্রাম বা সিস্টেমের ভুলত্রুটির কারণে ৫৯.৫ বিলিয়ন ডলার আর্থিক ক্ষতি হয়।

আমাদের অনেক সময় বাগ দূর করার জন্য বিভিন্ন ধরনের সফটওয়্যারের ওপর পুরোপুরি নির্ভর করা কোনো মতেই ঠিক নয়। আর এটি কখনই সবশেষ বা নির্ভরযোগ্য সমাধান হতে পারে না। বিশ্বখ্যাত কমপিউটার বিজ্ঞানী Dijkstra-র মতে, 'program testing can be used to show the presence of defects, but never their absence.' সুতরাং একথা নিশ্চিত করে বলা যায়, কোনো প্রোগ্রাম দিয়ে সফটওয়্যার পরীক্ষা-নিরীক্ষা করার পর এটি আপাতদৃষ্টিতে বাগমুক্ত মনে হলেও এর মধ্যে সমস্যা থাকবেই। একথা সত্য, অনেক প্রতিষ্ঠানই তাদের প্রোগ্রামিংয়ে বাগ দূর করার জন্য সফটওয়্যারনির্ভর পরীক্ষা-নিরীক্ষাকে অত্যধিক গুরুত্ব দিয়ে থাকে। এমনকি এসব প্রতিষ্ঠান কোডিংয়ের বাগ দূর করার জন্য যে পরিমাণ অর্থ খরচ করে থাকে তা সফটওয়্যারের তৈরির খরচের প্রায় অর্ধেক। সত্যিকার এসব পরীক্ষা-নিরীক্ষার ওপর নির্ভর না করে প্রথম থেকেই সফটওয়্যারের পরীক্ষণ, পর্যালোচনা ও ডিবাগিংয়ের দিকটি খুব গুরুত্বসহকারে খেয়াল রাখা দরকার।

তাহলে কী সফটওয়্যার টেস্টিংয়ের পদ্ধতির ওপর অধিক নির্ভর করা কোনোমতেই ঠিক নয়? একটি বড় ধরনের সফটওয়্যার স্বাভাবিক নিয়মে পরিপূর্ণভাবে পরীক্ষা করা কতটা সম্ভব? তাহলে কীভাবে সফটওয়্যারের বাগ সমস্যার সমাধান সম্ভব?

এ বিষয়ে বিশেষজ্ঞদের মতামত হচ্ছে, সফটওয়্যার বাগ সমস্যা দূর করার জন্য কোনো সফটওয়্যার টেস্টিং পদ্ধতির ওপর সম্পূর্ণ নির্ভর করা মোটেও উচিত নয়। বাগ সমস্যার অধিক ও কার্যকর ফল পাওয়ার জন্য কোড রিভিউ বা স্বাভাবিক পরীক্ষণ ও পর্যবেক্ষণ পদ্ধতি অনুসরণ করা সবচেয়ে ভালো। এ পদ্ধতি অনুসরণ করে ডেভেলপমেন্টের প্রতিটি স্তর যথা রিকোয়ারমেন্ট অ্যানালাইসিস, আর্কিটেকচার ডিজাইন, কোডিং প্রতিটি স্তর পর্যবেক্ষণ করতে হবে। এ ক্ষেত্রে একজন প্রোগ্রাম লিখবে, অন্যজন সেগুলো পর্যালোচনা করবে, যা বাগ দূর করার জন্য এখন পর্যন্ত সবচেয়ে নির্ভরযোগ্য পদ্ধতি। এই পদ্ধতিতেই সবচেয়ে বেশি সফলতা পাওয়া সম্ভব।

একথা সত্য, সম্পূর্ণ বাগমুক্ত কোনো সফটওয়্যার তৈরি করা অসম্ভব। তবে আমরা যেসব সফটওয়্যার ব্যবহার করে থাকি তা মোটামুটি বাগমুক্ত বলে ধরে নেয়া হয়। ■

তথ্যসূত্র : ইন্টারনেট

ফিডব্যাক : *animesh@tetbd.com*