Travel and Daily Allowance Management System

Nikhil Srivastava (B16CS020)

Zaid Khan (B16CS040)

Key Features Of The Software

- Enables a new user to register and login to the system.
- Felicitates the applying for reimbursement process.
- Reimbursement process broken down into two phases :
  - Before going for the trip.
  - After returning from the trip.
- Admin has the authority to approve or reject a reimbursement notification.
- A user can view his own notifications whereas the admin has the power to view all notifications.
- To integrate security features, proper security questions have been implemented to avoid breach of privacy

List Of Use Cases

- Login
- View Details
- Register a new user
- Forgot Password
- Change Password
- Authentication
- Apply For Reimbursement
- Approval From Admin
- Notification Updates

Database Structure (Files)

- Separate Files have been created for different users, to store their personal details.
- Files have been made to store all distinct reimbursement requests.
- A Username Database file has been made, to prevent reuse of an already in use username.
- Data in the file has been stored in a .csv (similar, because we used `, instead of ,) format, to ease out retrieval and edit routines from files.
- File Naming has been done in an easy to retrieve way.

Things we couldn’t implement?

- Couldn’t make use of effective and efficient Databases Already existing (SQLite, MongoDB) .
- Currently each student has to apply separately for reimbursement requests. We couldn’t implement a mentor feature, wherein the students could have chosen a mentor, and the mentor would have applied for all such students.
- Couldn’t use graphics.h library for effective use on console

Test Plan and Test Analysis

Test Plan:

1. Unit testing is performed for all modules described above in which the functionalities are not dependent on other modules.
2. For unit testing all modules described above in scope, Control flow testing and Data flow testing is applied which includes predicate coverage and complete branch coverage.
3. As all the modules are not independent and they have a shared interface and have a interdependency So attempts are carried out to perform appropriate system integration testing by using bottom up approach. All these are done appropriately to minimize errors.

Unit Testing:

Serial No.	**Use Case Tested**	**Pass/Fail**	**Issues**
1	RegisterManager: Register	Pass	No Issues
2	LoginManager: Login	Pass	No Issues
3	AuthenticationManager: Authentication Details	Pass	No Issues
4	AuthenticationManager: AuthenticateChangePassword Response	Pass	No Issues

Unit Testing:

Serial No.	Use Case Tested	Pass/Fail	Issues
5	Application: ApplyForReimbursement	Pass	No Issues
6	Notification: sendNotification	Pass	No Issues
7	DataBaseAccessLayer: fetchNotification	Pass	No Issues

System Testing:

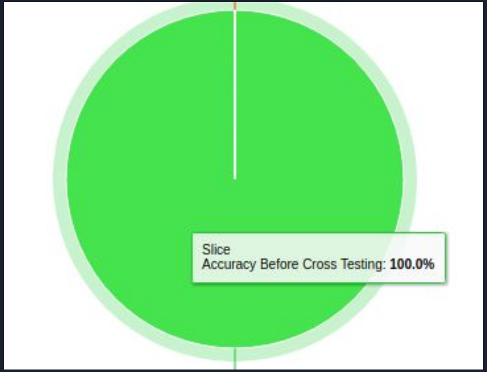
Serial No.	Use Case Tested	Pass/Fail	Issues
1	LoginAsUser->ApplyForReimburse-mement	Pass	No Issues
2	LginAsuser->CheckNotification	Pass	No Issues
3	LoginAsuser->ApplyForReimbursement (Previous Request)	Pass	No Issues
4	Register as student	Pass	No Issues
5	Register as Professor	Pass	No Issues

System Testing:

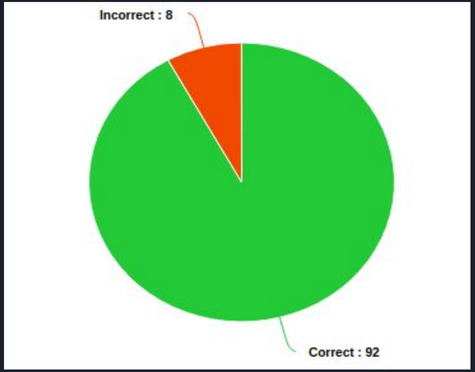
Serial No.	Use Case Tested	Pass/Fail	Issues
6	LoginAsAdmin->ShowAllNotifications	Pass	No Issues
7	ForgotPassword->LoginAsUser	Pass	No Issues

Overall Statistics

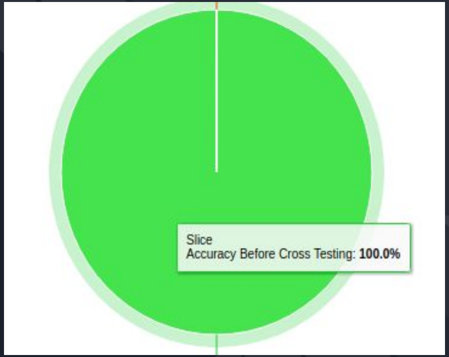
Accuracy before cross testing



Accuracy after Cross Testing



Accuracy After Enhancement (> 99%)



Testing Notes

- Unit Testing - CFG Used - Data Flow Testing and Control Flow Testing - Branch Coverage Method
- System Testing - Top Down Method and End to End Testing is done.
- Cross Project Tetsing - (3 errors found)
- During enhancement (All errors resolved)

Cross Testing Analysis

Attendance Management System

Group -18

List Of Use Cases Tested :

- Add Faculty()
- Add Student()
- Add Attendance()
- Modify Attendance()
- Login()

List Of Use Cases Tested :

- Out of 85 functions, we tested 10-12 use cases, which were enough to test all the use cases.
- Out of 12 use cases, 3 bugs were found.
- File Handling needs to re-looked into, while modifying user attendance

Conclusions :

- Use of DBMS over file handling can make the procedure much more efficient and effective.
- Valid changes from cross testing report were implemented in enhancement phase.
- C++ application required a lot of time and effort for checking errors during console interactions.
- Use of automated tools for testing could have made the testing more robust.

Thank You