# Student Monetary System

1. Kevin Mwenda - 101538
2. Jessie
3. Anthony Gitonga - 101985
4. Eric Burugu - 100036

# Process Description

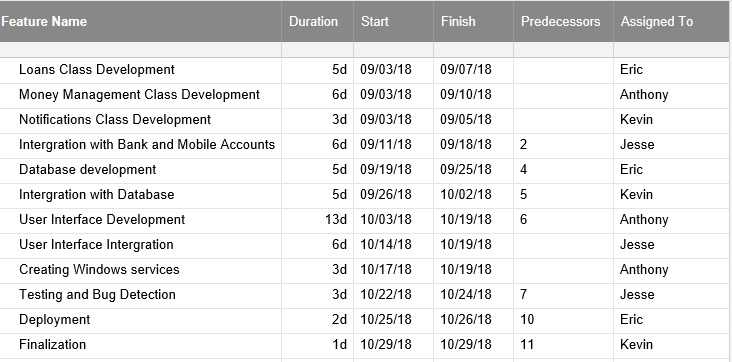
## Software Toolset

Our student banking application will be centered on providing a system that can help students manage their money and provide loans to them. For this to be accomplished, certain set of tools shall be used in development.

The main front and back-end programming language will be Java. Java is suitable since all members are adapt in it and it is known for its security. The source for our data will be students. The application will be centered on the students who will create accounts with us and be able to provide us with their information, so we can process it and help them in their money management. Since the application will be built by a team, we will need to use a version control system. It has been agreed upon that GitHub will be our version control system. All developers have their accounts and a repository has already been created by the group leader. The bug tracking tool to be used will be Bugzilla which is a free web-based bug tracking tool. We decided on it because it has an optimized database which ensures performance and scalability. It also has really good security. Trello will be used for assigning tasks between member. The application to be used for development will be NetBeans since it provides GitHub support and GitHub is our main version control system.

## Schedule/Timeline

It is estimated that the application will take about 9 weeks to be built. The schedule is as described below.



## Risk Summary

The major risks of developing this software are:

1. Deploying the database to an online cloud platform so that any the app can be used from anywhere. The security of the database is also a major issue.
2. Transaction and concurrency are also an issue since many users will be accessing the database at the same time.
3. Integration with banks and mobile accounts like Mpesa may not be possible if APIs are not available for that.
4. Time may be an issue, but division of labor will work in our favor.

The most worrying risk is the Integration with user accounts. If this is not possible we will have to ask users to update their spending every day and make the application into a budgetary app which will monitor all spending. This will also be the case for users who for one reason or another do not want to link their accounts to our system.

We will also research different cloud providers for database and choose the best that suits our needs. This will prevent the database from being unreachable remotely. Use of synchronized methods and sessions will also reduce risk of concurrency issues.

If time becomes an issue, the notification feature may be removed and that means that service classes will also be removed. This will mean that users will not get unsynchronized alerts in case of deadlines to pay loans or in case their prior set usage limit is exceeded.