***Degree Plan Automation***

***Team Deliverable-1***

***By Team Shan***

Nanditha Bodanapu

Sharanya Gottimukkula

Sriharshini Vallabhaneni

Aravind Thottempudi

**Project Description\*:**

As per the university rule, if a student has completed 18 credit hours he/she is required to submit a degree plan.

So far the manual process is as follows:

**Step 1:** The student has to choose a professor to be his/her advisor.

**Step 2:** If the professor accepts to be the student’s advisor.

**Step 3:** The student submits a degree plan form which has to be approved by the advisor.

**Step 4:** The professor has to manually check if the student has fulfilled the requirements like if he/she has listed all core subjects required for her plan etc. and approve/reject the form.

**Step 5:** Once the professor approves the plan the student must submit the approved form to Administrative specialist.

**Step 6:** The Administrative Specialist then has to forward those forms to Associate chair for approval.

**Step 7:** Once the degree plan is approved by the Associate chair, it has to be approved by the Department chair.

**Step 8:** A copy of degree plan would be sent to the student for reference.

**Step 9:** If the degree plan is rejected in any of step 4, step 5, step 6 and step 7 then the process must be repeated from step 3 till step 7.

As the process is manual the validations which are performed may not be accurate and may lead to resubmission of the degree plan iterative and time consuming,

Our goal is to automate the above process and make it seamless for the students as well as the professors.

The process will be as follows.

Our project will have two views/modules one for student and another for professor.

**Student view/Module:**

**Step 1:** Once the student completes 18 credit hours the student can register to the online portal.

**Step 2:** Once he/she registers, the student can choose a professor to be his/her advisor.

**Step 3:** The request would be sent to the respective professor on her behalf, the student can also fill in few comments while sending the request.

**Step 4:** The professor response would be sent to the student, If the professor rejects the request, then the student must repeat the previous step to a different professor until one of the professor approves.

**Step 5:** Once the professor approves the request, the student can fill in the degree form and send it to the professor.

**Step 6:** The professor can approve or disapprove it, he can also mention the reason why he disapproved it.

**Step 7:** If disapproved than the student has to resubmit it.

**Step 8:** Once it is approved by the professor, a copy would sent to the student as well as department.

**Professor View/Module:**

**Step 1:** The professor can register through registration page.

**Step 2:** The professor can choose to accept or reject students.

**Step 3:** The professor can keep track of accepted students.

**Step 4:** The professor can accept or reject degree plans.

**Step 5:** The professor can specify the reason for rejection of degree plan

**Software Development tools:**

**Programming Language:** Java 8, HTML 5, CSS 3, MySQL (Workbench)

**IDE:** Eclipse (photon)

**Framework:** Spring MVC (version 4)

**Software management system:** Maven, GitHub

**Server:** Apache Tomcat Server (version 9)

**Project Management Software:** Gantt Project

**Testing software:** Junit

**Hardware:** Windows 64 bit

**Plan for Project Timeline and expected milestones:**

The Work breakdown structure (WBS) for this project has 7 major steps that are the 7 milestones to be achieved, each of the steps have a certain set of activities to be accomplished by an established deadline. The project plan is according to incremental model of software development. The table below shows the steps, activities and their timelines.

|  |  |  |
| --- | --- | --- |
| **Task** | **Begin Date** | **End Date** |
| **1.0** Project Planning | 8/30/18 | 9/10/18 |
| 1.1 Project Specific Discussion | 8/30/18 | 8/30/18 |
| 1.2 Deciding Dev Environments | 8/31/18 | 8/31/18 |
| 1.3 Group Repository | 9/3/18 | 9/3/18 |
| 1.4 Project Timeline Planning | 9/4/18 | 9/5/18 |
| 1.5 Project Risks | 9/6/18 | 9/6/18 |
| 1.6 GIT Group Policy | 9/6/18 | 9/6/18 |
| 1.7 Project Planning Document | 9/6/18 | 9/10/18 |
| 1.8 End of Project Planning Phase | 9/11/18 | 9/11/18 |
| **2.0** System Design | 9/12/18 | 10/1/18 |
| 2.1 High Level System Design | 9/12/18 | 9/18/18 |
| 2.2 Software Design & Hardware Analysis | 9/19/18 | 9/24/18 |
| 2.3 User Interface | 9/24/18 | 9/28/18 |
| 2.4 Database Design | 9/24/18 | 9/28/18 |
| 2.5 Designing development Phases | 9/28/18 | 10/1/18 |
| 2.6 End Of Design Phase | 10/2/18 | 10/2/18 |
| **3.0** Development Phase 1 | 10/4/18 | 10/23/18 |
| 3.1 User Interface | 10/4/18 | 10/18/18 |
| 3.2 Code + Unit Tests | 10/8/18 | 10/19/18 |
| 3.3 Database Implementation | 10/8/18 | 10/19/18 |
| 3.4 Integration | 10/22/18 | 10/23/18 |
| 3.5 End of Development Phase 1 | 10/24/18 | 10/24/18 |

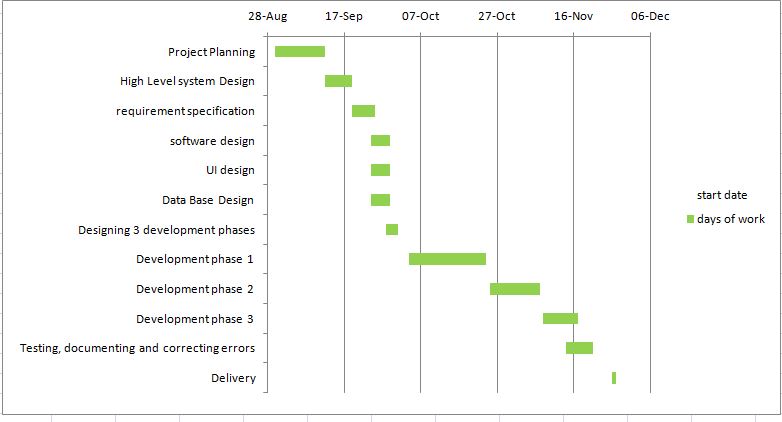
|  |  |  |
| --- | --- | --- |
| 4.0 Development Phase 2 | 10/25/18 | 11/6/18 |
| 4.1 Analyzing Additional Requirements | 10/25/18 | 10/26/18 |
| 4.2 Code + Unit Tests | 10/26/18 | 11/5/18 |
| 4.3 Integration | 11/6/18 | 11/6/18 |
| 4.4 End Of Development Phase 2 | 11/7/18 | 11/7/18 |
| 5.0 Development Phase 3 | 11/8/18 | 11/19/18 |
| 5.1 Analysis of New requirements | 11/8/18 | 11/9/18 |
| 5.2 User Interface | 11/12/18 | 11/15/18 |
| 5.3 Code + Unit Tests | 11/14/18 | 11/19/18 |
| 5.4 End Of Development Phase 3 | 11/20/18 | 11/20/18 |
| 6.0 Testing | 11/20/18 | 11/23/18 |
| 6.1 System Tests | 11/20/18 | 11/23/18 |
| 6.2 Documentation Of Identified Errors | 11/20/18 | 11/22/18 |
| 6.3 Code Fix For Identified errors | 11/21/18 | 11/23/18 |
| 6.4 Retesting Updated Code | 11/21/18 | 11/23/18 |
| 6.5 End Of Testing Phase | 11/26/18 | 11/26/18 |
| 7.0 Product delivery | 11/26/18 | 11/26/18 |

**Gantt Chart (Detailed):**

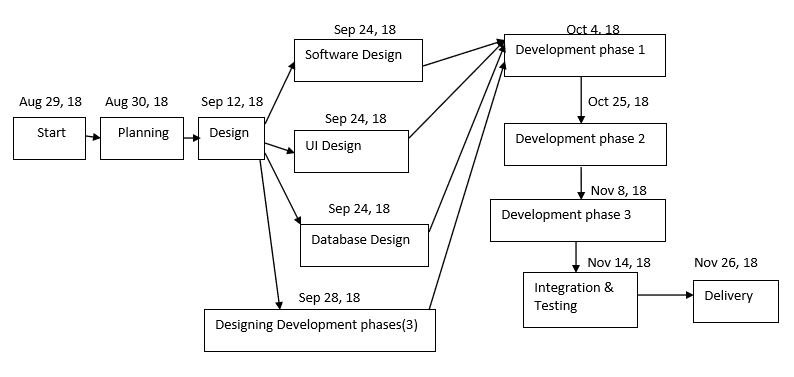
The PDF below has the Gantt chart with detailed schedules for each of the activities and the deadlines of the achievable milestones.



**Gantt Chart (outline):**



**Pert Chart:**



**Risk Management:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Risk Type** | **Description** | **Corrective Measure** | **Risk Type** |
| 1 | Communication Gaps | - Lack of proper communication may affect the project development. | **-** Regular meet ups on Tuesdays, Thursdays and Fridays to ensure that we have full clarity of what we are doing | Generic |
| 2 | Unexpected Delays in Schedule | - Sometimes due to some unexpected technical difficulties we may not be able to stick to the schedule  - Absence of team member due to inevitable(valid) reasons. | - Will monitor the schedule on a regular basis  - Will ask the rest of the team members or the responsible team member to spend extra time on the project to compensate for the lost time | Generic |
| 3 | Lack of Familiarity with Spring MVC | -Project is based on Spring MVC  -Only one resource has experience with Spring MVC | - Dedicated extra hours to learn Spring MVC model  - Dedicated one day to receive knowledge transfer from experienced resource | Specific |
| 4 | Maintenance of coding standards and efficiency | - Writing the code just to show up the functionality to the user without bothering about the efficiency and standards will affect the overall performance of the application.  -Only two resources have professional experience, hence have knowledge of coding standards and code efficiency which will be a considerable risk as far as the application performance is considered | - One experienced resource paired with another resource  - Also we will attend a training to know more about the coding standards and efficiency  -Peer reviews | Specific |
| 5 | Mapping of values from Frontend to Backend | - The development for frontend and backend would be performed simultaneously as per our project schedule.  - Improper Mapping of values from frontend to backend may  cause severe setback for our project. | -While designing the modules, we would make sure that the variables are correctly mapped from frontend to backend | Specific |

**Team member roles:**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Role** | **Responsibilities** |
| Nanditha Bodanapu | Backend  Java Developer | -Backend design and development Student module.  -Peer reviewer for Professor module.  -Responsible for updating meeting minutes every week in Git. |
| Sharanya Gottimukkula | Frontend Developer and Database Design | -Database design and Integration.  -Peer reviewer for front end module. |
| Sriharshini Vallabhaneni | Frontend Developer | -Frontend design and development.  -Peer reviewer for database design. |
| Aravind Thottempudi | Backend  Java Developer | -Backend design and development Professor module.  -Peer reviewer for Student module. |

**Project repository checkout and update policies:**

To avoid conflicts our team has come up with the following update policies

**1.** Every team member will create their own branch to work.

**2.** The developer can commit and push the code to the head branch after successful compilation and developers test (where applicable).

**3.** Before pushing to Master, the developer must pull changes from master to their local branch, check and resolve conflicts (if any) and then push the code changes to their branch head.

**4.** The developer must use proper commit messages describing the changes/updates implemented in code.

**5.** Every Sunday the developer can push code from their local branch to their respective Branch head.

**6.** From Sunday till next Friday the latest code changes need to be reviewed and approved by a peer reviewer, the changes (if any) must be made and committed by Friday morning.

**7.** Once all the latest changes are in branch head all the code changes are completed. The branch head can be pushed to master by Friday evening.

**References:**

[1] Software Engineering: A Practitioners Approach, 8th Edition, Roger Pressman and Bruce Maxim, McGraw-Hill

[2] https://ourcodingclub.github.io/2017/02/27/git.html

[3] Referred Lecture notes by Hyunsook Do