**D424 – Software Capstone**

**Task 2**



|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | Student Degree Planner |
| **Student Name:** | Galen Hammerle |

[**Business Problem** 3](#_Toc205108177)

[**The Customer** 3](#_Toc205108178)

[**Business Case** 3](#_Toc205108179)

[**Fulfillment** 3](#_Toc205108180)

[**SDLC Methodology** 3](#_Toc205108181)

[**Deliverables** 3](#_Toc205108182)

[**Project Deliverables** 3](#_Toc205108183)

[**Product Deliverables** 4](#_Toc205108184)

[**Deployment Plan and Outcomes** 4](#_Toc205108185)

[**Project Timeline** 4](#_Toc205108186)

[**Environments and Costs** 5](#_Toc205108187)

[**Programming Environment** 5](#_Toc205108188)

[**Environment Costs** 6](#_Toc205108189)

[**Human Resource Requirements** 6](#_Toc205108190)

[**Validation and Verification** 7](#_Toc205108191)

[**Sources** 9](#_Toc205108192)

# **Business Problem**

**The Customer**

The target audience of this application is university students. They must follow a prescribed academic plan to achieve their educational goals while managing multiple responsibilities outside of school. The application helps them keep track of scheduled education events (e.g., upcoming courses and assessment notifications). By centralizing academic planning, the app supports students in reaching their educational milestones.

## **Business Case**

The application offers value to universities and students. University students utilize this mobile application to track past terms and courses, while receiving notifications about upcoming courses and assessments. These reminders help students plan for academic success. As a result, improved student performance can contribute to higher retention rates, enhanced institutional reputation, and increased enrollment.

## **Fulfillment**

A mobile application, built using Microsoft’s cross-platform .NET MAUI framework, will deliver the required functionality. This includes a login screen, and pages for students to view terms, courses, and assessments. Courses and assessments will have notification capability to alert students to upcoming deadlines. Course notes can be created and shared. Students can shift required courses between terms to suit their needs. Application users with an administrative account can create, update or remove users, courses, terms, and assessments. Data will be saved in an SQLite database. Role-based authorization will be used to enforce data security.

# **SDLC Methodology**

This project uses an incremental SDLC Methodology with V-Model inspired test paring. This methodology will be used due to well defined requirements and testing as each feature is added to the project. The V-Model builds quality into the system early. For example, during the development of the log-in page, related tests will be developed, these will be functional tests or unit tests where suitable. Tests developed for each feature will be documented and utilized at later portions of the project. Bugs found during the testing phases will be documented and resolved before moving on to the next feature. Testing will be performed by the development team. When all features have been completed, including feature testing, a regression test of all features will occur prior to deployment.

# **Deliverables**

## **Project Deliverables**

* Project Schedule
  + Week 1: Define new feature workflow and requirements, captured in a document.
  + Week 2: A low fidelity wireframe will be created using for new and existing pages.
  + Week 3: Create a database relationship diagram outlining all features.
  + Week 4: Implement database schema and create mock data.
  + Week 5: Implement mock data and the Log-in screen.
  + Week 6: Enforce user role permissions.
  + Week 7: Final testing will be conducted followed by deployment of the application.
* Test Plans
  + Each feature will have a defined test plan used to evaluate completion. All feature test plans will be incorporated into a final test plan.
  + Test results will be captured in a test report for the feature including commit numbers of related fixes, and subsequent test results.

## **Product Deliverables**

Product Deliverables represents what is produced to deliver to the customer.

* User Manual
  + A document detailing how the application is used will be provided.
* The Application
  + The application will be released on the Google Play Store (or similar).

# **Deployment Plan and Outcomes**

The final product is a cross-platform student management application designed to help university students track academic progress, and plan for upcoming assessments. Users of the application will be provided with a secure download link which will include installation instructions.

An administrator will provide the user with appropriate login credentials. A comprehensive user manual will be made available alongside the application and source code in a private GitLab repository.

Users encountering issues or requesting enhancements should submit them through the GitLab tracking system. This ensures centralized management of bugs, feedback, and feature requests. The development team will monitor and provide timely feedback.

# **Project Timeline**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Phase | Milestone/Task | Resources | Dependencies | Deliverable | Description | Dates w/ Duration |
| Pre-development | Requirement Definition | Project Lead | None | Requirements | Define application use and feature flow | 9/1/2025 – 9/5/2025  5 days |
| Design | User Interface Design | UX\UI Team | Requirements Document | Low fidelity wireframe  High fidelity mockup | Create the UI that relates the look and feel of the project | 9/8/2025 – 9/12/2025 5 days |
| Design | Database Design | Database Team | Requirements Document | ERP Model | Relationship model of the database | 9/15/2025-9/19/2025  5 days |
| Implementation | Database implementation | Database Team | ERP Model | Mock Data csv files, unit tests, code loading mock data | The application shall create and load new tables and load mock data. Unit tests will validate the implementation. | 9/22/2025-9/26/2025  5 days |
| Implementation | Create login screen | Development Team | Database implementation and mock data | Functional user log in screen data. Data loads to term, course, instruction, and notification pages. | A user log in screen will be added to the application. User will login to access content. | 9/29/2025-10/1/2025  3 days |
| Implementation | Test login screen | QA Team | Functional login screen. | Test Plan and test results. | Create and execute user log in test plan. | 10/2/2025-10/3/2025  2 days |
| Implementation | Enforce User Roles | Development Team | User log in screen  Requirements Document | Application with user roles implemented according to the requirements document. | Implement user role restrictions. | 10/6/2025-  10/8/2025  3 days |
| Implementation | Test User Roles | QA Team | Application with enforced user roles | User Role Test plan and results. | Create and execute test plan for user requirements. | 10/9/2025-10/10/2025  2 days |
| Regression Test | Perform Regression Testing | QA Team | Feature complete application  User Log In test plan  Application test plan | Regression Test results. | Perform regression testing and document results. | 10/20/2025-10/24/2025  5 days |
| Deployment | Documentation | Technical Writer | Regression tested application  Regression test plan | User manual in docx and pdf format. | Update and publish user manual. | 10/27/2025-10/28/2025  2 days |
| Deployment | Application Link | Development Team | User Manual  Application installer | Accessible web link to the application. | Publish application and documentation. | 10/29/2025  1 day |

# **Environments and Costs**

## **Programming Environment**

The following tools, languages, and frameworks were selected for application development:

* **Tools**: Visual Studio 2022, Git, GitLab
* **Languages**: C#, XAML
* **Frameworks**: .NET MAUI, SQLite

.NET MAUI was selected as the core framework for this application due to its cross-platform capabilities, which allow development for both Android and Windows. Our team has over a decade of experience using Visual Studio IDEs to build C# WPF applications, which rely on XAML—a markup language also used by .NET MAUI. This familiarity enables efficient reuse of existing skills.

SQLite was chosen as the local database solution due to its lightweight, serverless nature and compatibility with mobile platforms. GitLab provides a cloud-hosted Git repository with issue tracking and version control tools aligned with modern development workflows.

## **Environment Costs**

Visual Studio Community is a free integrated development environment provided by Microsoft (Microsoft, n.d.-a). The development of this application qualifies under the terms outlined in the Visual Studio Community License (Microsoft, n.d.-b). The .NET MAUI framework, C#, XAML, and mobile emulators are included in the Visual Studio installation at no additional cost.

GitLab offers a free tier suitable for individual developers and small teams, allowing up to five licensed users and private repositories (GitLab, n.d.). SQLite is an open-source database engine that integrates seamlessly with .NET MAUI applications (SQLite, n.d.).

## **Human Resource Requirements**

This project is scheduled for seven weeks. Labor is estimated at $120 per hour, a rate that falls near the midpoint of the $90–$160 hourly range for U.S.-based software developers (FullStack Labs, 2023). Billable time is estimated at six hours per day, accounting for non-project-related activities such as administrative tasks or meetings.

**Estimated Cost Calculation**:  
7 weeks × 5 days/week × 6 billable hours/day = **210 billable hours**  
210 hours × $120/hour = **$25,200** estimated labor cost

# **Validation and Verification**

This application will undergo multiple phases of testing, including unit testing, functional testing, regression testing, and user acceptance testing. Testing will follow an iterative cycle: test → identify issues → fix → retest. The goal is to ensure the application is stable, functional, and meets user requirements prior to deployment.

**Unit and Functional Testing**

Unit tests will be created and executed using mock data to validate individual components of the application. Functional tests will be developed to ensure that each feature performs according to the specification. Initial test plans will be created by the QA team and updated as features are implemented. All tests and test results will be documented using Microsoft Word.

**Bug Tracking and Resolution**

When a defect is found, the QA team will log the issue in GitLab's issue tracker, identifying the failed test conditions. When an issue is entered into GitLab the development team will be notified, the issue will be prioritized and assigned to a developer. The assigned developer(s) will then investigate the issue, determine the root cause, and implement a fix. A short description of the root cause and solution will be added to the Release Notes in GitLab. Fixes will be committed to source control (Git) and retested using the updated test plan. Each fix will be tracked in GitLab, and progress will be recorded in the associated test documentation.

**Regression Testing**

Once all core features are implemented and tested, full regression testing will be performed to ensure changes have not introduced new issues. This will include executing all test cases across all major workflows of the application. Regression test results will be documented and used to determine readiness for release. Regressions found during testing will be documented according to the bug tracking process. Once all regression issues have been resolved another round of regression testing will begin.

**User Acceptance Testing (UAT)**

After regression testing, the application will undergo user acceptance testing. The product management team will use the application in a controlled environment to validate that it meets user needs and business requirements. Feedback from UAT will be reviewed by the QA and development teams to determine if any final adjustments are needed.

**Post-Deployment Testing**

After deployment, the application will be tested on end-user devices to confirm installation works as expected and that the application runs reliably in its intended environment.

**Justification**

This testing strategy ensures that defects are identified early, traced to their source, and resolved efficiently. Using GitLab for issue tracking ensures accountability and allows traceability from test plan to resolution. Maintaining test artifacts in Word documents allows for clear audit trails, especially useful in environments where formal documentation is required. Multiple layers of testing, including UAT and post-deployment validation, ensure the final product is both technically sound and user-focused.

# **Sources**

1. **Visual Studio Pricing (Free Tools Tab)**  
   Microsoft. (n.d.-a). *Visual Studio pricing: Free tools*. Microsoft. Retrieved August 2, 2025, from <https://visualstudio.microsoft.com/vs/pricing/?tab=free-tools>
2. **Visual Studio Community License Terms**  
   Microsoft. (n.d.-b). *Visual Studio Community 2022 license terms*. Microsoft. Retrieved August 2, 2025, from <https://visualstudio.microsoft.com/license-terms/vs2022-ga-community/>
3. **GitLab Pricing  
   GitLab. (n.d.). *GitLab pricing*. GitLab. Retrieved August 2, 2025, from** [**https://about.gitlab.com/pricing/**](https://about.gitlab.com/pricing/)
4. **SQLite GitHub Repository**  
   SQLite. (n.d.). *sqlite/SQLite [Repository]*. GitHub. Retrieved August 2, 2025, from <https://github.com/sqlite/sqlite?tab=readme-ov-file#readme>
5. **FullStack Labs**

FullStack Labs. (2023, August 7). *Software development price guide: Hourly rate comparison*. Retrieved August 2, 2025, from <https://www.fullstack.com/labs/resources/blog/software-development-price-guide-hourly-rate-comparison>