

ADAPT Community Network Systems Design Document

August 27, 2025

Revision History

Date	Version	Author	Changes

Contents

SECTION 1

Project Overview.....	3
1.1 Project name and Description.....	4
1.2 Stakeholders.....	??
1.3 Assumptions.....	??
1.4 Core-features.....	??

SECTION 2

Functional Requirements.....	3
2.1 Performance.....	4
2.2 Scalability.....	??
2.3 Security.....	??
2.4 Reliability.....	??
2.5 Maintainability.....	??

SECTION 3

System Architecture.....	3
3.1 High-Level Diagram.....	4
3.2 Technology Stack.....	??
3.3 System Components.....	??

SECTION 4

Module Design.....	3
4.1 Authentication & Authorization Module.....	??
4.2 Class Management Module.....	4
4.3 Reporting Module.....	??
4.4 Attendance Recording Module.....	??

SECTION 5

Database Design.....3

5.1 ER Diagram.....4

5.2 Schema Design.....??

5.3 Indexes.....??

SECTION 6

API Design.....3

6.1 Endpoints (HTTP Method, Request/Response Format, etc).....4

6.2 Authentication.....??

6.3 Authorization (roles, resources, permissions).....??

6.4 Rate Limiting.....??

6.5 Error Handling.....??

SECTION 7

Security Design.....3

7.1 Authentication/Authorization.....4

7.2 Data Encryption.....??

7.3 Security Auditing.....??

7.4 Vulnerabilities.....??

7.5 Security Stack.....??

SECTION 8

Deployment Architecture.....3

8.1 Environment Setup (Development, Staging, Production).....??

8.2 Scaling Strategy.....??

8.3 Monitoring Stack.....??

SECTION 9

Testing Strategy.....3

9.1 Unit Testing.....??

9.2 Integration Testing.....??

9.3 Acceptance Testing.....??

9.4 Performance Testing.....??

9.5 Security Testing.....??

9.6 Automated Testing.....??

SECTION 10

Maintenance and Monitoring.....3

10.1 Logging.....??

10.2 Alerting.....??

10.3 System Health Montioring.....??

10.4 Error Tracking.....?

SECTION 11

Backup and Recovery.....3

11.1 Backup Strategy.....??

11.2 Disaster Recovery.....??

SECTION 12

Risks and Mitigation.....3

12.1 Technical Risk.....??

12.2 Mitigation Strategies.....??

SECTION 13

Future Enhancements.....3

13.1 Roadmap.....??

13.2 Scalability Considerations.....??

1 Project Overview

1.1 Project Name:

This project will be called the ADAPT Community “**Attendance Tracker**”, or just Attendance Tracker. For now, I will opt to keep it simple until a more creative or appropriate name is proposed.

1.2 Project Description:

This project was proposed and created with the goal of streamlining and consolidating the workflow ADAPT Community network specialists into one platform. Our responsibilities include the following:

1. Taking daily attendance for our classes
2. Adding new students to our classes
3. Creating and removing classes from our schedules
4. Submitting weekly schedules
5. Translating class and student data to pre-styled excel sheets

As of now, all of the responsibilities above must be done manually, which is a common source of stress for the specialists. At its completion, the project will allow for the automation for every single responsibility in one place.

1.3 Stakeholder:

This project was proposed by coworker Wilma Cox, and will be supported by Shaniece Frank, Peter Cobb, and Chevonne Brown.

1.4 Assumptions:

Initially, this project was started in May, with a release sometime in September. Unfortunately, due to hardware failure, and a priority on another ADAPT project for jeopardy score keeping, the completion date for the first version had to be pushed back. With a new laptop, and the scorekeeping project completed, there can now be clear timelines for each milestone. Moving forward, I will proceed with the following assumptions:

Resource Assumptions:

- Development will be handled primarily by me, with UI/UX and general aesthetics handled by Wilma Cox.
- Specialists will create test classes alongside test students for validation.
- Specialists will work within a stable and easy to use environment.

Technology Assumptions:

- For the back-end, all data processing will be done with ASP.NET, with views being handled By HTMX and Alpine.js for interactivity.
- Class and student data will be exported to the official specialist, pre-styled excel sheet.
- All class and student data will be managed by a relational database, most likely PostgreSQL.

Scope Assumptions:

- Version 1.0 will include the ability to add classes, students, and export to excel sheets.
- Additional features like the admin dashboard will be released with later versions of the app.

Risk Assumptions:

- No other projects will interfere with the development timeline of this project
- Staff will be able to start using it as soon as the app is released.

Dependency Assumptions:

- Specialists will access the application through modern browsers (Chrome, edge, Safari, etc).
- Application will be hosted on existing IT infrastructure.

Milestones:

- Version 1.0 release date - **10/1/25**
 - - Features: Add classes, Add students, export them to excel
- Version 1.1 release date - **11/1/25**
 - - Features: Admin page for supervisors to view specialists classes.

1.5 Core-features:

This project as mentioned before will consolidate every single clerical responsibility into a single app to reduce complexity and improve efficiency. Here is a summary of the features:

- Add classes
- Add Students
- Export classes and files to excel
- Admin page for supervisors to view and edit classes/schedules
- Create weekly schedules

2 Functional Requirements

2.1 Performance:

Response Times

- The ideal response times for adding new students, adding new classes, and adding students to classes should ideally fall within 1-2 seconds assuming a solid internet connection.
- Generating the excel spreadsheet that contains all of the specialists classes, and students who attended for a specific month should ideally take no longer than 5 seconds.

Throughput/Concurrency

- This application should be able to very easily handle the load of all of the specialists using it at the same time, as there are only around 25 of us with classes and students to add.
- Max usage for the app should be at the end of each day, and near to the deadline for the monthly attendance sheet submission.

Transaction Rates

- During peak periods when the app is being used the most, a good benchmark for the amount of transactions per minute is 100, combining classes and students being added.

Client-Side Performance

- For solid internet connections, classes and students for each class should all load in at most two seconds, ideally under one second.
- Front-end interactions should be instant, and not really too much on re-loading the page to fetch new information.

2.2 Scalability:

Due to the internal nature of this application, and the limited number of ADAPT specialists using the application at any given time, **vertical scaling** will be the preferred scaling method. Horizontal scaling would require deploying the source code on multiple machines, and I currently do not have access to such resources, but even if I did, it would be overkill for this project. With vertical scaling, I can deploy a single instance to a cloud provider, and focus entirely on expanding it should the need ever come up for my team.

2.3 Security:

For security, I plan to implement all of the best practices for protecting log in, as well as other methods to satisfy HIPAA requirements and protect information about people-supported. Methods include:

- Anti-CSRF token for log in.
- Passwords will be hashed before being stored in the database.
- Session usage on the backend to maintain login/logout flow.
- Refresh tokens to prevent users from getting signed out too quickly.
- Sign up restricted to users with @adaptn emails.
- HTTPS for all data transmission.
- 2FA using the authenticator app on mobile.
- Encryption for people-supported names for SQL database.

2.4 Reliability:

This app should be as reliable as possible, and in order to meet that threshold, I will guarantee the app will meet several benchmarks set by me.

Expected uptime:

Uptime is defined as the amount of time the website will actually be up and running, in contrast to downtime which is how long websites stay down. With this project being hosted on the free tier on the cloud service Azure, there will some down time as the website will “sleep” when not using , but it will only affect the first initial request, with each subsequent one operating normally.

Data Durability:

Database will be backup every other day, with timestamps marked to date each backup

Disaster Recovery:

- Disaster Recovery is defined as the ability for an application to recover valuable data should there ever be a breach, or some server related crisis that causes the application to shut down while in use. In such a scenario, there are some tools I have available to address them now, and better tools in the future to more effectively deal with it.
- As mentioned before, my cloud SQL database (Azure storage) will be backed up daily, with rollback to earlier versions possible should anything go wrong. Excel Attendance exports for each specialist can also function as back up data since it contains the classes and attendance marks for each student.
- In the case where the project has to be redeployed, moved to another cloud service, or even run from the source (this laptop) process will be made much simpler due to the application being containerized using docker.

Failover Systems:

- A failover describes the process of the primary server automatically switching to another backup server should the primary server fail. Due to this project being hosted on a single cloud service with a single instance, the system must be manually restarted through the cloud service provider.
- As the project grows and becomes more widely adopted, load balancing will be adopted to distribute client requests to different servers and instances rather than just one as I am currently doing now.

2.5 Maintainability:

Here is a list of active practices that will be done to keep this project as maintainable as possible:

Codebase Standards:

Every single file in this project will follow C# naming and styling conventions for clarity and consistency.

Razor pages will be organized according to standard project hierarchy.

For Class names, PascalCase will be used, and for variables, camelCase will be the default

Dependency Management:

Every single fil

Documentation:

Testing:

For the purposes of testing, unit testing and integration testing will both be used to ensure every aspect of the application has accounted for every single possible edge case.

- **Unit Testing** is a type of test that checks each individual method, function, class, etc in isolation to ensure the expected outcome is returned.
- **Integration Testing** is a type of test that checks to see if different layers of the program interact with each other as intended, and also to ensure database connectivity is functioning. This type of test would test to see if passwords, usernames, and classes for example are all actually being saved to the database.

Logging:

3 System Architecture

3.1 High-Level Diagram: Attendance