Curriculum Autumn 2017*

Bachelor in IT – Mobile Programming
Faculty of Technology

All information revised as of October 2016. Note that all curricula may be updated*
1 Introduction

The bachelor’s programme in Mobile Programming is technically oriented, anchored in front page technologies. At completion of the programme, students will be able to develop a mobile concept taking its starting point in the needs of a case or business concept. The students will know how to choose the most adequate platform (web or native) and to design the interface for a mobile solution and implement it. They will be able to actively facilitate client-server communication by creating services and interfaces that function as content suppliers in the mobile solution.

To achieve such qualifications, you will need knowledge about tools, and an introduction to the technologies. The programme Mobile Programming gives you the tools and knowledge you need, including a thorough introduction to interface design, script language and programming on Android and the Apple iOS platform. This knowledge prepares you to take the digital life of the future one step further. You will further be introduced to innovation and mobile ecosystems. This is knowledge that will make you able to facilitate information architecture for communication with the end user and to open for cooperation with mobile platforms.

1.1 Curriculum overview – Mobile Programming

The course structure of the programme is shown in the matrix below.

<table>
<thead>
<tr>
<th>Sem6</th>
<th>Research Methods (7.5 ECTS)</th>
<th>Bachelor’s Thesis (22.5 ECTS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sem5</td>
<td>Elective (7.5 ECTS)</td>
<td>iOS Programming (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Algorithms and Data Structures (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Web Development and API Design (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem4</td>
<td>Project Software Engineering (15 ECTS)</td>
<td>Innovation and Prototyping (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mobile Development (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cross Platform Development (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem3</td>
<td>Elective (7.5 ECTS)</td>
<td>Advanced Java Programming (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mobile Ecosystems (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interface Design (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem2</td>
<td>Elective (7.5 ECTS)</td>
<td>Object Oriented Programming 2 (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information Security (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iterative Web Project (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem1</td>
<td>Databases 1 (7.5 ECTS)</td>
<td>Object Oriented Programming 1 (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital Technology (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Creative Web Project (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td>7.5 ECTS credits</td>
<td>7.5 ECTS credits</td>
</tr>
<tr>
<td></td>
<td>7.5 ECTS credits</td>
<td>7.5 ECTS credits</td>
</tr>
<tr>
<td></td>
<td>7.5 ECTS credits</td>
<td>7.5 ECTS credits</td>
</tr>
</tbody>
</table>

Yellow boxes denote that the course is taught jointly with one or more other study programmes. Green boxes show where the electives are placed in the curriculum. In the 2nd semester students choose an elective from a pool of optional courses across faculties and programmes in addition to the electives offered by the Faculty of Technology.
Electives in the 2nd semester (Spring)
DS2100 Animation (7.5 ECTS)
DS2200 Digital Culture (7.5 ECTS)
PG2201 Unity Development (7.5 ECTS)

Electives in the 5th semester (Autumn), Bachelor in Mobile Programming
DS5300 Interaction Design 3 (7.5 ECTS)
PG3300 Software Design (7.5 ECTS)
PRO300 Virtual Reality Project (7.5 ECTS)

The study in mobile programming builds on the first-year study of Bachelor in IT. The total programme is a 3-year study awarding the degree Bachelor in IT at Westerdals Oslo ACT. The first year is taught jointly for all IT programmes and gives solid basic qualifications in programming, project work, system development, data technique, and databases.

The second year specialises in mobile solutions and mobile programming. Further, it introduces mobile ecosystems, interface design, advanced Java, and innovation and prototyping. A project is carried out over two semesters: Project Software Engineering, that gathers all the knowledge gained so far, to be used in this fairly extensive group-based project.

In the third year emphasis is on a total understanding of mobile programming and mobile architecture. Programming for iOS platforms is introduced as well as API design for interaction with the mobile applications. The understanding of the totality of mobile solutions is enhanced, namely: from idea to product, realised as web as well as mobile solutions. The third year also includes jointly taught courses that are core courses in the Bachelor in IT education: an introduction to research methods introducing quantitative as well as qualitative methods, and a bachelor’s thesis that takes place in a company with students working in teams.

1.2 About the programme

The bachelor’s programme in IT specialising in mobile programming aims to educate candidates that are qualified to develop mobile concepts with their basis in business needs, where an adequate platform (web or native) has to be chosen and an interface is to be designed and implemented in a mobile solution. The emphasis of the programme on close cooperation with the industry, working with practical work tasks in cross-disciplinary teams, gives the students essential experience in working with complex issues as well as a broad basis for success in their further careers, working with adaptation for client-server communication by creating services and interfaces that function as content suppliers for mobile solutions.

At successfully completed 3-year study candidates will be characterised by the following knowledge, skills and general competence:

Knowledge – the candidate
• has wide knowledge of mobile solutions, applications, architecture, central theories and issues, system development methods and tools or programming IDE
• knows the characteristics and paradigms of mobile development
• knows the mobile ecosystem
• is acquainted with research and development work within the field of programming
• is acquainted with research and development work in the field of mobile development

Skills – the candidate
• is able to use knowledge and relevant results from research and development work on practical and theoretical questions and make well-founded choices
• reflects on her/his own work and adjusts it under guidance
• masters relevant tools (for IDE, version control, project management and testing) and methods

General competence the candidate
• understands relevant academic and professional questions
• is able to design and implement a mobile solution from business concept to finished solution
• is able to exchange points of view and experiences with other people in the same field, thus contributing to the development of sound practice
• is acquainted with new thinking and innovation processes

1.3 Central themes
The bachelor’s programme in Mobile Programming has the following central themes and research anchoring:

Programming skills are central throughout the study, for client-server and for mobile solutions. The study programme develops an understanding of design and implementation of mobile solutions supported by back systems.

The research anchoring is connected with mobile solutions and modern web information architecture. The study programme works actively with business and industry and in particular with the consulting industry. This industry cooperates by giving guest lectures and workshops as an integrated part of the teaching.

2 Individual courses offered to exchange students Autumn 2017

This bachelor’s programme has recently been revised and the first year of the revised programme started Autumn 2015. The third year of study will be taught as of Autumn 2017.

The planned courses in the third year of study are described in the following sections but please note that they may be further developed or changed.
2.1 PG6300 Web Development and API Design

Norwegian name: Webutvikling og API-design

ECTS credits: 7.50

Area of study: Technology/IT

Language of instruction: English

Programme: Mandatory course in Bachelor in IT – Mobile Programming.

Required prerequisites: Basic knowledge of web development and mobile application development is expected

Recommended prerequisites: None

Semester: The course is taught in the 5th semester

Course leader: Tor-Morten Grønli

Course outline

The transition to MVC-based front-end frameworks like AngularJS can be seen as one of the greatest paradigm shifts in the history of web development. This course introduces good practice and various tools to be used in the development of web applications and Application Programming Interfaces (APIs) in this new paradigm.

Learning outcome

Knowledge: At completion of the course candidates will
- have gained good knowledge of the MEAN stack (MongoDB, Express, AngularJS, and Node.js)
- know how an Application Programming Interface (API) functions
- be able to explain the abstraction Representational State Transfer (REST), and when this should be used
- know WebSockets and when to use them
- know the mechanics of Cross-Origin Resource Sharing (CORS) and its importance for the development of an API that supports several types of clients
- be able to explain central characteristics of a document database such as MongoDB and how it differs from a relation database like MySQL
- know basic aspects of information security in connection with the development of an web application with an API

Skills: At completion of the course candidates will be able to
- design, implement and automate testing of a RESTful API that communicates over JSON with sufficient data security
- design, implement and automate testing of a web application that communicates with the API
- deploy the solution of a sky service such as DigitalOcean or Heroku
- quickly build and deploy one or more applications that use the API by means of the MEAN stack and a good tool chain

General competence: At completion of the course candidates will
- know central concepts related to the development of a RESTful API that supports various types of clients, and
- evaluate the need to secure information in an API or a web solution
Teaching and learning methods
Lectures, exercises, self-study, assignments

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 52 hours
Examination and preparing for the examination – 100 hours
Total recommended workload – 200 hours

Technology and tools
- MongoDB
- Express
- AngularJS
- Node.js
- JetBrains IntelliJ IDEA, WebStorm or equivalent for development with JavaScript and to a self-chosen mobile platform

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on an individual written examination consisting of an extensive examination paper containing assignments the candidate has worked with, and received supervision on, throughout the course. The specifications for submission of the examination paper are handed out 4 weeks before submission.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
See Learning outcome

2.2 PG5600 iOS Programming
Norwegian name: iOS-programmering
ECTS credits: 7.5
Area of study: Technology/IT
Language of instruction: English

Programme: Mandatory course for Bachelor in IT – Mobile Programming, optional for Bachelor in IT – Intelligent Systems

Required prerequisites: Advanced knowledge of programming (from 2nd year Bachelor in IT) or equivalent previous knowledge in object oriented programming

Recommended prerequisites: None

Semester: The course is taught in the 5th semester (Autumn)

Course leader: Tor-Morten Grønli

Course outline
The course gives an introduction to programming in Swift and the iOS platform. At completion of the course the students will be able to make applications that communicate over networks, store data locally, and use interface elements and patterns that go with the platform.

Learning outcome
Knowledge: At completion of the course candidates will
- have gained knowledge of the architecture of the iOS platform
- be able to describe the life cycle of an iOS application using text and drawing
- know the process of distributing applications
- know how applications are compiled on iOS
- be able to describe MVC, the observable and delegate pattern in context with iOS, using text and drawing

Skills: At completion of the course candidates will be able to
- program basic Swift, including
  - data types
  - use of foundation classes
  - control structures
  - use of object orientation
  - use of protocols
  - use of extensions
  - use of closures
  - error handling
- use iOS APlet actively when programming applications
- apply the usual GUI components to produce user interfaces
- use storyboards
- write automated texts
- make use of basic animations
- serialise, de-serialise and persist data
- programme to internet-based services
- account for and implement asynchronous architecture
- debug and deploy an application to simulator and unit

General competence: At completion of the course candidates will be able to
- propose and give reasons for the choice of architecture in an iOS application
- know when asynchronism is relevant
• consume a REST API

Teaching and learning methods
The course is taught with 12 lectures, each of about 2 hours, and 12 teacher-led exercises of about 2 hours each. The teacher-led exercises are not mandatory, but to obtain the competence goals the students are expected to complete exercises and additionally put in some extra effort if the teacher-led exercises are not sufficient.

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 110 hours
Examination and preparing for the examination – 42 hours
**Total recommended workload – 200 hours**

Technology and tools
iOS SDK (inkl Xcode)
iOS

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on an individual written examination (100%) in which candidates submit an extensive examination paper. The paper tests knowledge as well as skills. The assessment criteria for the examination paper are handed out together with the question paper 2-3 weeks before submission.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
See Learning outcome
2.3 PG4200 Algorithms and Data Structures

Norwegian name: Algoritmer og datastrukturer

ECTS credits: 7.5

Area of study: Technology/IT

Language of instruction: English

Programme: Mandatory course in Bachelor in IT – Intelligent Systems; Bachelor in IT – Mobile Programming; Bachelor in IT – Programming; Bachelor in IT – Game Programming

Required prerequisites: Successfully completed PGR100 Object Oriented Programming 1, PGR101 Object Oriented Programming 2, or equivalent previous knowledge

Recommended prerequisites: None

Semester: The course is taught in the 5th semester (Autumn)

Course leader: Lars Sydnes

Course outline
The course introduces algorithms and data structures, which are central in the implementation and design of effective data systems. Asymptotic analysis of worst-case resource use is emphasised, as well as central algorithms and data structures in connection with search and sorting. The course also introduces a number of graph algorithms.

Learning outcome
Knowledge: At completion of the course candidates will

- know central abstract data types such as lists, stacks, queues, priority queues, sets or collections, symbol maps, trees and graphs
- know the properties of central data structures such as arrays, linked lists, heaps, binary trees, search trees, balanced search trees, hash arrays, and graphs implemented by neighbours lists and matrices
- know central search algorithms such as linear search, binary search, search in binary search trees, search in hash arrays
- know central sorting algorithms such as Insertion sort, Selection sort, Bubble sort, Quicksort and Merge sort
- know methods for traversing of graphs, Dijkstra’s algorithm, the A* algorithm, and Prim’s algorithm
- know the structure, effect and use of recursive functions, including recursive traversing of trees and graphs, recursive search and sorting, and backtracking
- know about difficulties in connection with measuring running time for computer programs
- know basic concepts in the theory of computation complexity, included the use of O notations, tilde notations, NP completeness and reduction, the Travelling Salesman Problem, and the Knapsack Problem

Skills: At completion of the course candidates will be able to

- use the knowledge mentioned in the section above about knowledge to use existing libraries for algorithms and data structures in an adequate way
- implement known data structures such as array lists, linked lists, binary search trees, heaps, hash arrays, and graphs
• implement known algorithms such as Insertion sort, Selection sort, Quicksort and Merge sort
• judge worst-case resource use for concrete computer programs by means of O notation and tilde notation
• master basic generic programming in Java
• compare empirical measurements of resource use with estimates derived from theory, as a coarse-grained evaluation of the estimates and as a method to estimate the size of unknown factors in the estimates

General competence: At completion of the course candidates will
• master classical asymptotic analysis of computer programs
• be able to use existing libraries and self-developed algorithms and data structures in a good way in their own programming practice
• master a language suited for discussions of resource use in computer programs
• have gained the knowledge of algorithms and data structures that is required in postgraduate studies of informatics

Teaching and learning methods
The teaching takes place as lectures and self-studies with guidance from the lecturer and student assistant teacher.

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 128 hours
Examination and preparing for the examination – 24 hours
Total recommended workload – 200 hours

Technology and tools
Access to pc and Java Development Kit 1.8

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
Requirements for taking the examination are that the candidate has submitted and received approval on 2 out of a total of 3 mandatory assignments.

Assessment
Assessment is based on an individual written examination of 3 hours. No aids permitted.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
See Learning outcome
2.4 DS5300 Interaksjonsdesign 3

Norwegian name: Interaksjonsdesign 3
ECTS credits: 7.5
Area of study: Technology/IT
Language of instruction: English
Programme: Mandatory course in Bachelor in IT – Interactive Design, and in Bachelor in IT – Mobile Programming
Required prerequisites: Successful completion of DS4300 or equivalent previous knowledge
Recommended prerequisites: None
Semester: The course is taught in the 5th semester
Course leader: Sturla Bakke

Course outline
In this course the students will gain advanced knowledge of theories related to interaction design and the use of interaction design patterns. The course takes the form of a design process in which the students learn step by step in the process.

Learning outcome
Knowledge: At completion of the course candidates will
• have gained knowledge of principles of interaction design
• have deep knowledge of usability and user testing
• have advanced knowledge of information architecture
Skills: At completion of the course candidates will be able to
• carry out data compilation and data analysis at different stages of a design process
• apply and evaluate design patterns
• apply principles of interaction design
• use personas and scenarios as tools in a design process
General competence: At completion of the course candidates will be able to
• use the gained knowledge of HCI (Human-Computer Interaction) in a design process

Teaching and learning methods
The course is taught with lectures, lab-based instruction, and individual and group-based project work.

Recommended workload
Participation in lectures and tutorials – 40 hours
Self-study – 60 hours
Independent preparation for presentations and discussions in class – 10 hours
Examination and preparing for the examination – 90 hours
Total recommended workload – 200 hours

Technology and tools
Tools for wireframes: Balsamiq in online version and Axure RP on lab machines
Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
The course is being revised and course requirements have not yet been decided. The decision is expected in the autumn of 2016.

Assessment
The course is being revised and the examination form grading scale have not yet been decided. The decision is expected in the autumn of 2016.

Assessment criteria
See Learning outcome

Notes
Westerdals Oslo ACT carries out an extensive revision of all study programmes in 2016. This revision of the course will be completed in the autumn of 2016. In particular, the examination form and the relationship between content, progression and examination are being thoroughly revised.

2.5 PRO300 Virtual Reality Project

Norwegian name: Virtual Reality-prosjekt
ECTS credits: 7.5
Area of study: Technology/IT
Language of instruction: English
Programme: Optional course in Bachelor in IT – Programming; Bachelor in IT – Game Programming; Bachelor in IT – Game Design; Bachelor in IT – 3D Graphics; Bachelor in IT – Interactive Design; Bachelor in IT – Intelligent Systems

Required prerequisites: Basic qualifications in programming, interaction, game or concept development, or 3D design
Recommended prerequisites: As above
Semester: The course is taught in the 5th semester (Autumn)
Course leader: Kim Baumann Larsen
Course outline
The Virtual Reality Project is a cross-disciplinary course where students participate, bringing with them their different backgrounds. The purpose of the project is to develop interactive virtual reality applications for games, art or visualisation. The students will make concepts, design and create interactive Virtual Reality solutions, and thereby understand the technological frames in use.

Learning outcome
Knowledge: At completion of the course candidates will
- know the machine and software platforms for Virtual Reality
- know the platform requirements for different Virtual Reality solutions
- know the possibilities and limitations with Virtual Reality
- know the demands for achieving the largest possible presence in a Virtual Reality solution
- know basic principles for storytelling for interactive Virtual Reality

Skills: At completion of the course candidates will be able to
- draft and plan one interactive Virtual Reality concept for games, art or visualisation
- make prototypes for different platforms
- design and implement one interactive Virtual Reality experience for a chosen platform
- integrate 3D models, sound elements and interaction points into a holistic Virtual Reality experience for a chosen platform

General competence: At completion of the course candidates will be able to
- assess and create an optimal Virtual Reality solution for a chosen platform
- assess different Virtual Reality platforms and the possibility and limitations of solutions

Teaching and learning methods
Lectures and exercises, and one cross-disciplinary project in which students participate with their different study programme backgrounds either in programming, interaction, games/concept development, or 3D design.

Recommended workload
Participation in lectures and tutorials – 25 hours
Self-study – 80 hours
Independent preparation for presentations or discussions in class – 5 hours
Student work with projects, productions, assignments etc. – 20 hours
Independent exercises, lab work, practical work individually or in groups – 70 hours
Total recommended workload – 200 hours

Technology and tools
HTML editor

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.
Coursework requirements
None

Assessment
Assessment is based on a combined examination consisting of the following elements:

• A group project examination (60%) where the groups submit the results of the project they have been working with in the course
• A group written examination (20%) consisting of a group report describing how the project was carried out, as well as reflections in connection with the work (a process document). The report should have a length of 3000-5000 words. The specifications of requirements for the report are handed out three weeks before submission.
• A group oral examination (20%) in which the groups present the project result and the process. The oral examination lasts approx. 25 minutes.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

2.6 PG3300 Software Design

Norwegian name: Software Design
ECTS credits: 75
Area of study: Technology/IT
Language of instruction: English
Programme: Mandatory for Bachelor in IT – Programming and Bachelor in IT – Game Programming; optional course in Bachelor in IT – Intelligent Systems

Required prerequisites: Experience with basic object oriented programming, such as PGR100 Programming 1 and PGR101 Programming 2, or equivalent knowledge

Recommended prerequisites: None
Semester: The course is taught in the 3rd or 5th semester (Autumn)
Course leader: Tomas Sandnes

Course outline
The course will make students able to design and further develop extensive software systems using well-known techniques for modelling, testing and implementation.

Learning outcome
Knowledge: At completion of the course candidates will

• know the background and content for the UML standard
• know what Unit Testing is
• be able to explain the principles of test-driven development (TDD)
• know par programming and how the use of it influences software projects
• know what design patterns are
• know and be able to identify some important design patterns when reading them in code
• know what refactoring is
• know what multi-threading is
understand how locking can be used to code thread securely
be able to explain the principles of event handling

Skills: At completion of the course candidates will be able to
• know and use the basic syntax in the programming language C#, and know in what ways it is different from Java
• handle the UML diagrams: use case, class diagram, and sequence diagram
• use UML to design program architecture
• participate productively in pair programming
• implement the following patterns: MVC & MVP, singleton, factory, builder, flyweight, composition, decorator
• use design guidelines such as layers and the GRASP principles emphasising the following for the latter: controller, information, expert, low coupling, high cohesion
• implement applications that apply several threads
• master an integrated development tool (IDE)
• write and edit source code with the mentioned tool

General competence: At completion of the course candidates will be able to
• reflect on multi threading and its usage
• cooperate with other programmers to develop good programs and to further develop their competence
• assess the quality of existing programs and possible structural improvements

Teaching and learning methods
Lectures, exercises, self-study

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 104 hours
Examination and preparing for the examination – 48 hours
Total recommended workload – 200 hours

Technology and tools
IDE: Visual Studio

Learning method/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on a two-part examination consisting of the following elements:
• A written examination in groups (40%) for which the candidates submit self-developed program code and an accompanying document. The assignment has a time span of 3 weeks.
• An individual oral examination (60%) of 20 minutes. The oral examination takes its point of departure in the above-mentioned work (the written examination), and is used to decide an individual final grade adjusted according to the individual candidate’s performance as shown in the oral examination situation.

Assessment criteria
See Learning outcome